VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Pohybové senzory - jejich použití pro ovládání počítače lidským tělem

Motion Sensors - their use for Controlling of the Computer by Human Body

2014 Bc. Petr Hlavinka

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Petr Hlavinka**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Pohybové senzory - jejich použití pro ovládání počítače lidským tělem
Motion Sensors - their use for Controlling of the Computer by Human Body

Zásady pro vypracování:

Cílem práce je prostudovat možnosti zařízení Kinect, frameworky od firmy Microsoft a případně další možnosti pro jiné operační systémy. Další části práce je rozšíření již implementovaného skeletonu, zaměření se na detekci různých gest a vývoj vlastní aplikace.

Jednotlivé body zadání:
1. Popis technologie Kinect se zaměřením na práci s gesty, seznámení se s novým framworkem firmy Microsoft.
2. Použít navrženou architekturu frameworku pro usnadnění práce s gesty směrem k aplikacím pro hendikepované.
3. Implementace rozšíření frameworku a testovací aplikace.
4. Propojení vytvořených aplikací s projektem Edukin
5. Otestování aplikace jako celku včetně jejího nasazení.

Seznam doporučené odborné literatury:

[1] Diplomová práce V. Bojka: Infrastruktura založená na Windows Azure pro projekt Edukin
[2] Diplomová práce A. Kašpara: Framework pro práci s gesty založený na technologii Kinect

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Mgr. Pavla Dráždilová, Ph.D.**

Datum zadání: 01.09.2013

Datum odevzdání: 07.05.2014

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA * Fakulta elektrotechniky a informatiky * OSTRAVA 1

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.*

V Ostravě 5. května 2014 .....

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 5. května 2014 .....

Rád bych na tomto místě poděkoval paní Mgr. Pavle Dráždilové za odbornou pomoc a konzultaci, bez které by tato práce nevznikla.

Abstrakt

Tato práce je zaměřena na vytvoření aplikace pro hendikepované osoby s využitím pohybového senzoru Kinect od firmy Microsoft. Hlavním cílem je navrhnout způsob, jak úspěšně detekovat gesta, která nejsou součástí Kinect SDK. Aplikace funguje online i offline, s využitím databázového serveru, sloužícího pro správu uživatelských účtů, ukládání výsledků a nastavení. Tato hra je začleněna do projektu Edukin a slouží pro procvičení rozpoznávacích a paměťových schopností a také rozvoj jemné a hrubé motoriky.

Klíčová slova: C#, Kinect, Edukin, WPF, Pohybové senzory, Gesta

Abstract

This thesis aims to create an application for disabled persons using the motion sensor Kinect from Microsoft. The main objective is to propose a way to successfully detect gestures, that are not part of the Kinect SDK. Application works both online and offline, using a database server to manage user accounts, storing results and settings. This game is incorporated into the project Edukin and is used for practicing cognitive and memory skills and also fine and gross motor skills.

Keywords: C#, Kinect, Edukin, WPF, Motion sensors, Gestures

Seznam použitých zkratek a symbolů

API	–	Application Interface
C++	–	C Plus Plus
C#	–	C Sharp
CRUD	–	Create Read Update Delete
DPI	–	Dots Per Inch
Edukin	–	Educaiton with Kinect
FPS	–	Frames Per Second
HD	–	High Definition
IR	–	Infrared
MB	–	Megabyte
RGB	–	Red, Green, Blue
SDK	–	Software Develompent Kit
SQL	–	Structured Query Language
WPF	–	Windows Presentation Foundation
XAML	–	Extensible Application Markup Language

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Již v době vzniku prvních počítačů bylo nutné vymyslet, jakým způsobem bude člověk s počítačem komunikovat. Asi nikoho nepřekvapí, že vývoj klasické počítačové klávesnice byl inspirován obyčejným psacím strojem. Jednalo se o velmi efektivní způsob, jakým může člověk obsluhovat a řídit činnost počítače. Revoluci v tomto směru nastolil Douglas Engelbart, který roku 1963 vytvořil první počítačovou myš [1]. Název myš byl odvozen od drátu, který připomínal myší ocásek. Myš měla pouze jedno tlačítko. Tělo bylo vyrobeno ze dřeva a snímání pohybu bylo realizováno pomocí dvou kovových disků, které na sebe byly kolmé, tudíž zaznamenávaly pohyb v obou směrech. Z dnešního pohledu je to úsměvné, jelikož můžeme běžně koupit myš s mnoha nejrůznějšími tlačítky. Snímače v těchto myších jsou schopny snímat s rozlišením jednotek tisíc dpi.

S postupným vývojem nových zařízení a objevování jejich využití, začali lidé přemýšlet nad vytvářením nových způsobů interakcí člověka. Jako jeden z velmi úspěšných se ukázal přímý kontakt se zobrazovanou plochou. Dotykové displeje dnes můžeme vidět na většině zařízení, jako jsou tablety a mobilní telefony. Poskytují dostatečnou míru přesnosti a jejich hlavní výhodou je, že displej je rovnou i komunikačním rozhraním, tudíž není třeba hardwarové klávesnice, která jinak zabírá další místo. Dotykové obrazovky jsou vhodné u mobilních zařízení, avšak pro vývojářskou práci se už tolik nehodí. Zde s přehledem stále vyhrává klávesnice a myš. Důvodem je zcela jistě přesnost těchto rozhraní. I tak je dotyková obrazovka velmi zajímavým způsobem interakce a ve sféře mobilních zařízení je velice populární.

Zatím nejzajímavějším prostředkem, jakým může člověk ovládat různá zařízení je přirozený pohyb. Již pár let na trhu můžeme vidět zařízení jako je Nintendo Wii Remote, Sony Playstation Move či Microsoft Kinect. Všechna tato zařízení snímají pohyb, i když fungují na zcela jiném principu. K ovládání u systému Wii a Move je potřeba ovladač, který funguje jako „ukazatel"v prostoru. Microsoft Kinect pracuje bez ovladače, ovladačem je sám člověk, což Microsoft podtrhl reklamní kampaní „You are the controller" [2]. Všechny tyto systémy jsou používány u herních konzolí těchto firem, jimiž jsou Nintendo Wii, Microsoft Xbox 360 nebo Sony Playstation 3. I když je Kinect velmi zajímavým zařízením, naráží na své limity. Ty jsou však posunuty zase trochu dále díky novému Kinectu, který již není pouze doplňkem, nýbrž je dodáván přímo jako součást konzole Xbox One. Všechny tyto výše uvedené systémy jsou komerčně dostupné, otázkou však zůstává, zda již existují armádní systémy či technologie, které dokáží zprostředkovat interakci mezi uživatelem na ještě lepší úrovni.

Cílem této práce je využít možností nabízených senzorem Kinect a vytvořit aplikaci, která se začlení do seznamu aplikací spadajících pod projekt Edukin. Hlavním důvodem vytvoření této aplikace je větší míra pohybu hendikepovaných pacientů, procvičení jejich svalů, rozvoj hrubé či jemné motoriky. V neposlední řadě budeme schopni díky analýze dat získaných během používání aplikace zjistit, zda se pacient k aplikaci vrací a dělá pokroky. Určitá skupina pacientů, kteří disponují mentálním postižením, postrádá smysl pro pohyb. Virtuální hry jsou jedním ze způsobů, který dokáže tuto skupinu lidí oslovit. Vytvořená zábavná aplikace by měla dokázat vzbudit u cílové skupiny zájem. Jelikož

je obsluha aplikace realizována pohybem, musí pacienti zapojit své vlastní tělo, což je žádaný výsledek. Interakci v této aplikaci bude zajišťovat pohybový senzor Kinect od firmy Microsoft.

Dále tato práce obsahuje přehled o tom, na jakém principu Kinect funguje a co stálo za jeho vznikem. Vysvětlíme si, jaká data nám Kinect poskytuje a k čemu slouží. Také se podíváme na infrastrukturu projektu Edukin, včetně ukázky již hotových aplikací. V kapitole Oblékání s Kinectem bude vysvětleno, jaká je myšlenka vytvořené hry, jakým způsobem probíhá ovládání a co je jejím cílem. V dalších kapitolách bude popsána samotná implementace aplikace včetně použitých technologií. Na konci práce bude provedeno zhodnocení jednotlivých metod pro detekci požadovaných gest.

2 Kinect

Roku 2005 na Tokyo Game Show předvedla firma Nintendo poprvé prototyp bezdrátového pohybového ovladače. O rok později Nintendo představilo dokončenou konzoli s názvem Wii. Tu bylo možné ovládat pomocí ovladače Wii Remote. Ovladač je schopen manipulovat s předměty pouhým namířením na obrazovku a navíc dokáže snímat pohyb ve všech třech osách. Mezi lidmi se stala tato konzole velmi žádanou. Do dubna roku 2011 se prodalo více jak 86 milionů kusů [3]. Díky skvělým prodejům a velké popularitě se jednalo o nejúspěšnější herní konzoli firmy Nintendo.

2.1 Historie

Divize Xbox firmy Microsoft v čele s Petrem Moorem se nechala inspirovat myšlenkou bezdrátového ovládání konzole Xbox a to jen pomocí vlastního těla. Zařízení, s kterým měl Microsoft přijít na trh, mělo být ještě mnohem lepší než Wii. Účelem bylo rozšířit možnosti herní konzole Xbox 360. Microsoft tedy začal s vývojem vlastního bezdrátového pohybového senzoru.

Vedoucí interního oddělení Xbox Alex Kipman se setkal se zakladateli firmy Prime Sense, která se specializuje na výrobu senzorů transformujících snímanou scénu do tří dimenzionálního prostoru. Aby byly šance na úspěch vyšší, Microsoft založil dva týmy, které pracovaly s odlišnou technologii na rozpoznání hloubky obrazu. První tým pracoval s technologií od firmy Prime Sense, zatímco druhý s technologií vyvinutou firmou 3DV. Cíl pro oba týmy byl jasný. Do výstavy E3 v roce 2007 předvést to nejlepší co se oběma skupinám podařilo vytvořit. Bohužel se vývoj potýkal s problémy. Navíc divizi Xbox opustil vedoucí Petr Moore, který odešel pracovat do Electronic Arts. Tuto pozici po něm převzal Don Matrick, který roku 2008 oživil celý projekt zajímavým videem o fungování technologie Prime Sense. Koncept ovládání se Mirosoftu zalíbil a celý projekt dostal zelenou s limitem pro dokončení projektu před Vánoci roku 2010. Do té doby vzniklo přes 70 aplikací a her využívajících Kinect, tudíž v době nasazení do prodeje již zákazníci nemuseli čekat dlouhé měsíce na jejich vývoj.

Interně bylo zařízení označováno kódovým jménem Project Natal. Microsoft pro něj vymyslel název Kinect, který je odvozen od slova kinetický. Úspěch Kinectu byl obrovský. Za prvních 60 dnů prodeje Microsoft prodal přes 8 milionů kůsů, a Kinect se tak zapsal do Guinessovy knihy rekordů jako nejrychleji prodávaná spotřební elektronika [7].

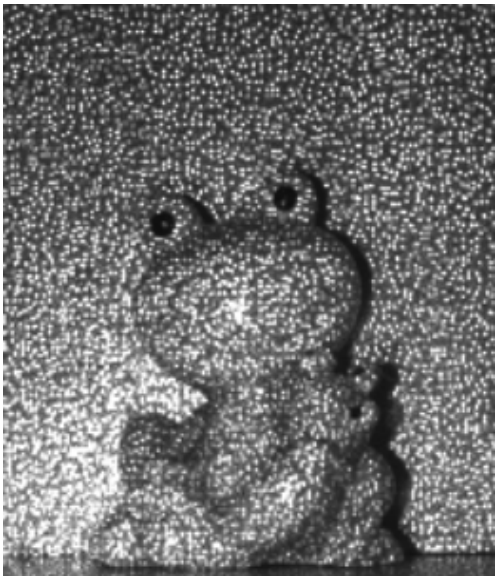
V únoru roku 2012 byla vypuštěna nová verze s názvem Kinect pro Windows, která je primárně určena vývojářům desktopových aplikací s možností programovat aplikace v jazyce C++, C#, či Visual Basic. Co se týká specifikací, jsou stejné až na jednu výjimku. Hloubkový senzor u verze Kinect pro Xbox dokáže pracovat v rozmezí 800 až 4000 mm. Nová revize přinesla v tomto směru změnu. Senzor podporuje takzvaný blízký mód, díky kterému je možné snímat objekty v rozmezí 500 - 3000 mm. Navíc byla přidána podpora sledování člověka vsedě.

2.2 Technologie

Jak již bylo řečeno v kapitole 2.1, Microsoft vyvíjel koncept založený na obou technolgiích. I když ve finálním produktu technologie 3DV není použita, Microsoft tuto firmu zakoupil. Hlavním důvodem akvizice bylo vyhnutí se potencionálním patentovým sporům spjatých se společným vývojem. Prototyp zařízení obsahoval snímač a čip od firmy Prime Sense, konkrétně čip PS1080, který je zodpovědný za zpracování barevného a hloubkového obrazu s rychlostí 30 snímků za vteřinu. Hlavní výhodou čipu PS1080 oproti technologii 3DV je jeho nízká cena.

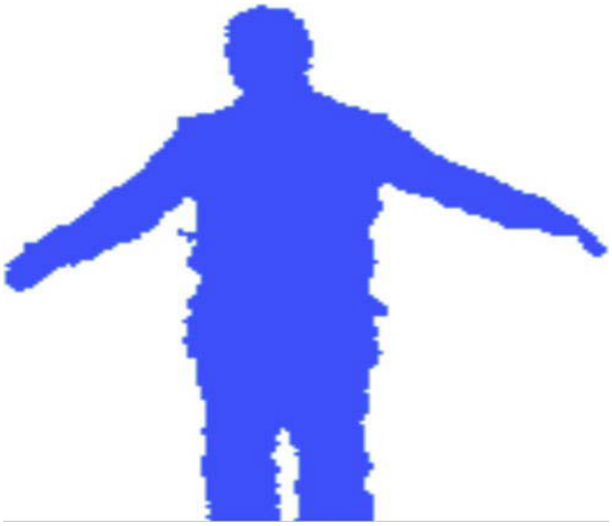
Získávání hloubkových dat z prostoru u technologie 3DV je založeno na vysílání paprsků do prostoru. Paprsek se odráží od plochy, na kterou dopadne a putuje zpět do senzoru. Hloubku obrazu pak lze změřit měřením času mezi jeho vysláním a příjmem. Princip technologie PrimeSense je zcela jiný. Infračervený projektor vysílá do prostoru stovky malých infračervených teček rozložených do určitého vzoru. Tyto infračervené body jsou pak snímány infračerveným senzorem. Tyto body můžeme vidět níže na obrázku 1.

Jelikož jsou objekty zakřiveny, zakřiví se podle nich i jednotlivé vzory. Data z infračerveného snímače pak putují do čipu a ten z nich vytvoří hloubkovou mapu. Ta je pak mapována a zarovnána vůči barevnému obrazu. Zarovnání se musí provádět z jednoho prostého důvodu. Klasická RGB a infračervená kamera jsou od sebe posunuty o několik centimetrů, tudíž obrazy nejsou identické [8]. Aby bylo snímání objektu spolehlivé, doporučuje se nevystavovat snímané objekty přímému slunečnímu záření. Můžeme si také všimnout, že na pravé straně snímaného objektu nejsme schopni získat data díky vrhání stínů a úhlu snímání.



Obrázek 1: Pohled na objekt pomocí infračervené kamery

Metoda získávání hloubkových dat již byla hotova. Nyní Microsoft stál před důležitým úkolem, a to vymyslet způsob, jak správně sledovat pohyb uživatele. Raná verze firmwaru vyžadovala, aby se uživatel postavil do polohy T, díky které byl sledovací software schopen uživatele zachytit a zaměřit se na něj. Bohužel se velmi často stávalo, že software ztrácel informaci o tom, kde se uživatel nachází. Aby došlo ke znovu nalezení uživatele, musel se uživatel opět postavit do polohy T, kterou můžeme vidět níže na obrázku 2.



Obrázek 2: Vyfiltrovaná hloubková mapa [7]

Z hloubkové mapy je možné jednoduše získat pixely, které reprezentují tělo uživatele. Bohužel bez segmentace jednotlivých částí nemáme žádné informace o tom, v jaké relaci jsou jednotlivé části lidského těla. Díky spolupráci pánů Fitzgibbona and Blaka vymyslel pan Shotton algoritmus, který dokázal rozdělit tělo na 31 částí [4]. To bylo demonstrováno na E3 v roce 2009, kdy bylo zobrazeno celkem 48 bodů kostry z rekonstruovaného lidského těla. Dnešní Kinect SDK používá pouze 20 z nich. Aby se neopakovaly problémy s nalezením těla, využili inženýři sílu samoučení. Vývojářský team dostal k dispozici stovky snímků s lidmi v různých polohách, včetně materiálů z hollywoodských nahrávacích studií, která měla k dispozici mnoho snímků tančících, běžících či jinak postavených lidí. K analýze těchto dat bylo v Microsoft Resarch rozhodnuto využít distribuovaného systému Dryad, který byl schopen na základě dodaných dat vybudovat rozhodovací strom. Ten rozděluje části těla hráče do jednotlivých částí. Tyto rozhodovací stromy jsou dnes součástí oficiálního SDK [7].

Microsoft do zařízení ještě zabudoval sadu čtyř mikrofonů, které se starají o záznam zvuku. Díky tomu je možné zadávat hlasové příkazy, s kterými má již Microsoft zkušenosti z předchozích let díky rozpoznávání řeči v operačních systémech Windows.

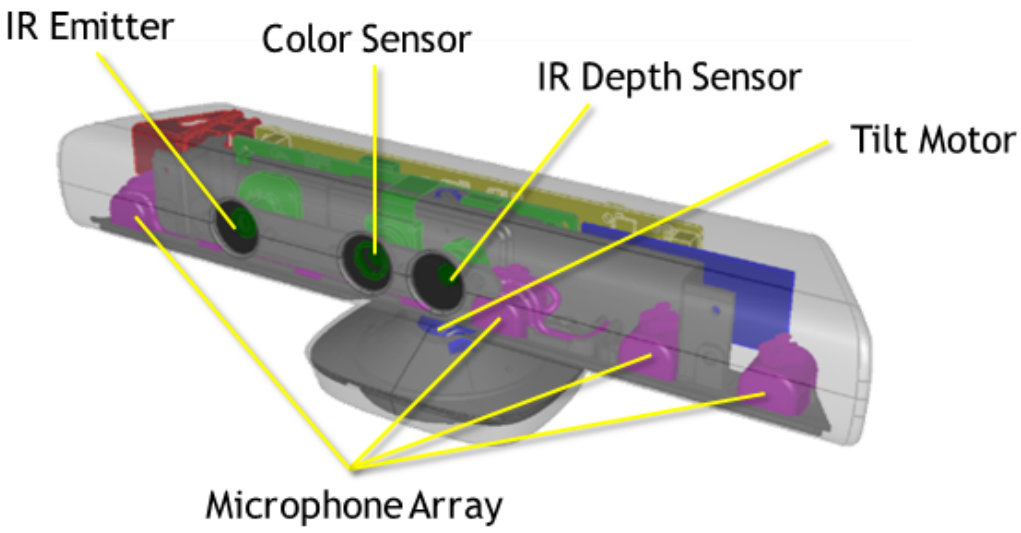
Obrázek 3: Rozdělení lidského těla na 31 částí [7]

2.3 Kinect a hardware

Vlastní hardware Kinectu je složen z několika částí. Převážnou část tvoří senzory a kamery, dodávané firmou Prime Sense. Mezi hlavou a stojánkem se nachází elektrický motorek, který je určen k naklápění zařízení. Microsoft silně doporučuje nepřetěžovat motorek častým naklápěním. Aby bylo zabráněno poškození motorku ze strany špatného zacházení, je omezen počet změn na jednu za sekundu. Po 15 změnách je nutné počkat minimálně 20 vteřin, než je možno znovu naklápění použít.

Jak můžeme vidět na obrázku 4, Kinect je složen z několika hlavních částí:

- Infračervený vysílač vytvářející zdroj infračerveného záření.
- RGB kamera, produkující obrazová data s rozlišením 1280*960 pixelů.
- Infračervený senzor s rozlišením 640*480 pixelů.
- Elektrický motorek určený k naklápění hlavy.
- Sada čtyř mikrofonů pro snímání zvuku, díky použití více mikrofonů je software schopen detekovat směr a polohu zdroje zvuku.



Obrázek 4: Hardware Kinectu [9]

Kinect	**Specifikace**
Zorné pole snímačů	43° vertikálně, 57° horizontálně
Vertikální náklon hlavy	$\pm$ 27°
Snímkovací frekvence barevného a hloubkového streamu	30 FPS
Akcelerometr	2G/4G/8G akcelerometr s přesností na 1°

Tabulka 1: Specifikace Kinectu

Zde můžeme vidět minimální hardwarové požadavky: [10]

- Dvoujádrový (x86) nebo x(64) bitový procesor o frekvenci alespoň 2.66 GHz
- USB 2.0 rozhraní
- 2 GB operační paměti
- Grafická karta podporující DirectX 9.0c

2.4 Kinect SDK

Již při samotném vydání Kinectu pro Xbox začalo spoustu nadšenců zkoumat možnosti zařízení. Bylo vytvořeno několik frameworků na získávání dat z Kinectu. Populárním se stal libfreenect vyvíjený komunitou OpenKinect a OpenNI vyvinutý přímo dodavateli hardwaru firmou PrimeSense. Půl roku po vydání v červenci roku 2011 Microsoft vypustil oficiální SDK nástroje pro práci s Kinectem. Díky této prodlevě byl často kritizován za

špatné zhodnocení situace. Microsoft mohl očekávat enormní zájem o tuto novinku ze strany technologických nadšenců a vědců z celého světa.

I když je vytvořené SDK primárně určeno pro zařízení Kinect pro Windows, je kompatibilní i se starší verzí Kinect pro Xbox. Aktuální verze SDK 1.8 umožňuje sledovat najednou až 6 lidí přičemž vytvářet kostru dokáže pro dva z nich. Kostru je možné reprodukovat pomocí 20 hlavních bodů. Dále je dostupné rozpoznání hlasu již ve 20 jazycích. Čeština však bohužel podporována není. Pokud použijeme zařízení Kinect pro Windows, SDK umožňuje použít blízký mód, který byl zmíněn již v kapitole 2.1. Nová verze Kinectu podporuje sledování člověka sedícího na židli. V tomto režimu dokáže software reprodukovat celkem 10 bodů kostry.

Propojení Kinectu s počítačem je realizováno pomocí rozhraní USB. Koncovka však není standardní a je k ní potřeba redukce na klasické USB. Co se týká propojení, narazil jsem na jeden nedostatek. Kinect je sice schopen komunikovat i přes USB 3.0, avšak streamy z RGB a hloubkové kamery jsou náhodně pozastavovány a to až na několik sekund. Ovladače jsou tedy plně funkční výhradně s USB 2.0, což by mohl být problém u nových základních desek, které již mají jen USB 3.0 rozhraní. Během léta 2014 se očekává vypuštění SDK ve verzi 2.0, které by mělo být připraveno pro práci s novým Kinectem, který je součástí konzole Xbox One.

V poslední verzi SDK přibyla sada nových knihoven, zjednodušujících inicializaci zařízení. Do aplikací mohou vývojáři nyní použít komponentu Kinect Sensor Chooser, která zobrazuje uživateli informace o stavu Kinectu. Vývojářům je poskytnuta sada metod pro práci s touto komponentou. Díky tomu jsme schopni reagovat na změny při připojení, ztrátě napájení či odpojení Kinectu. Aplikaci pak můžeme pozastavit na základě těchto podnětů.

2.5 Streamy

Data, která nám Kinect poskytuje, můžeme rozdělit do několika hlavních streamů. Těmi jsou `ColorImageStream`, `DepthImageStream` a `SkeletonStream`. Každý tento stream produkuje jiná data.

Práci se streamy můžeme rozvrhnout do tří fází. Nejdříve musíme daný stream povolit. Následně se zaregistrujeme na událost vytvoření nového snímku, která je vyvolána ve chvíli, kdy jsou data z daného streamu připravená. Nakonec zkopírujeme získaná data a můžeme s nimi dále pracovat. V závislosti na požadavcích naší aplikace můžeme jednotlivým streamům nastavit jejich formát a to zcela jednoduše pomocí enumu, který je vstupním parametrem funkce pro povolení konkrétního streamu. Například u barevného streamu jsme schopni nastavit barevný formát, rozlišení či snímkovací frekvenci.

ColorImageStream

Tento stream obsahuje data z RGB kamery. Často je využíván vývojáři k vytváření zábavných fotografií během hraní. Dále jej také můžeme použít pro analýzu obrazu nebo odfiltrování pozadí od hráče na základě hloubkových dat. Pixelům hráče pak můžeme ponechat RGB složku, zatímco zbytek scény zprůhledníme, tudíž docílíme efektu používaného ve filmech se zeleným plátnem.

DepthImageStream

Díky tomuto streamu jsme schopní získat data o hloubce každého pixelu v obrazu. Každý pixel je reprezentován jako 16 bitová hodnota, kde nejlevějších 13 bitů určuje hloubku daného pixelu v milimetrech. Zbylé 3 bity jsou rezervovány pro index hráče. Tento index jednoznačně určuje, které hloubkové pixely patří kterému hráči. SDK je schopno rozlišit celkem 6 hráčů na základě hloubkové mapy.

SkeletonStream

Aby bylo možné detekovat postavení hráče, potřebujeme vědět, v jaké relaci se nachází jednotlivé význačné body hráčova těla. V SDK tyto body vystupují pod názvem joint. Těch obsahuje kostra celkem 20. Jednotlivé jointy můžeme vidět na obrázku 5. Díky nízkému rozlišení a rušení se však může stát, že jednotlivé jointy mohou trpět takzvaným třepáním anglicky známé pod názvem jittering. Abychom tomuto negativnímu jevu zamezili, můžeme využít třídu `TransformSmoothParameters`. V této třídě nastavíme jednotlivé parametry jako je automatická korekce polohy, zjemnění pohybu, rádius korekce, predikce či maximální odchylka. Díky tomu je poté pohyb kostry plynulý a lokální odchylky jednotlivých jointů jsou kompenzovány. Doporučené hodnoty se neudávají, jelikož záleží na požadavcích konkrétní aplikace. SDK je schopno reprodukovat kostru pro dva hráče. Abychom od sebe mohli kostry rozlišit, obsahují objekty `Skeleton` jednoznačný identifikátor `TrackingId`.

2.6 Rozšíření streamů

V posledních verzích Kinect SDK Microsoft přidal dvě velmi zajímavé rozšíření stávajících streamů. Těmi jsou stream pro odfiltrování pozadí od uživatele a interakční stream pro detekci stavu ruky uživatele. Poněkud zarážející je fakt, že knihovny pro práci s těmito streamy nejsou přímo součástí základního SDK, ale jsou dostupné pouze jako přídavný balíček nazvaný Kinect for Windows Toolkit. Bohužel k nim neexistuje ze strany Microsoftu dokumentace. Stejně tak nejsou obsaženy v základním jmenném prostoru `Microsoft.Kinect`. Pokud chceme tyto streamy používat, je nutné přiložit k spustitelnému souboru další knihovny.

BackgroundRemovedColorStream

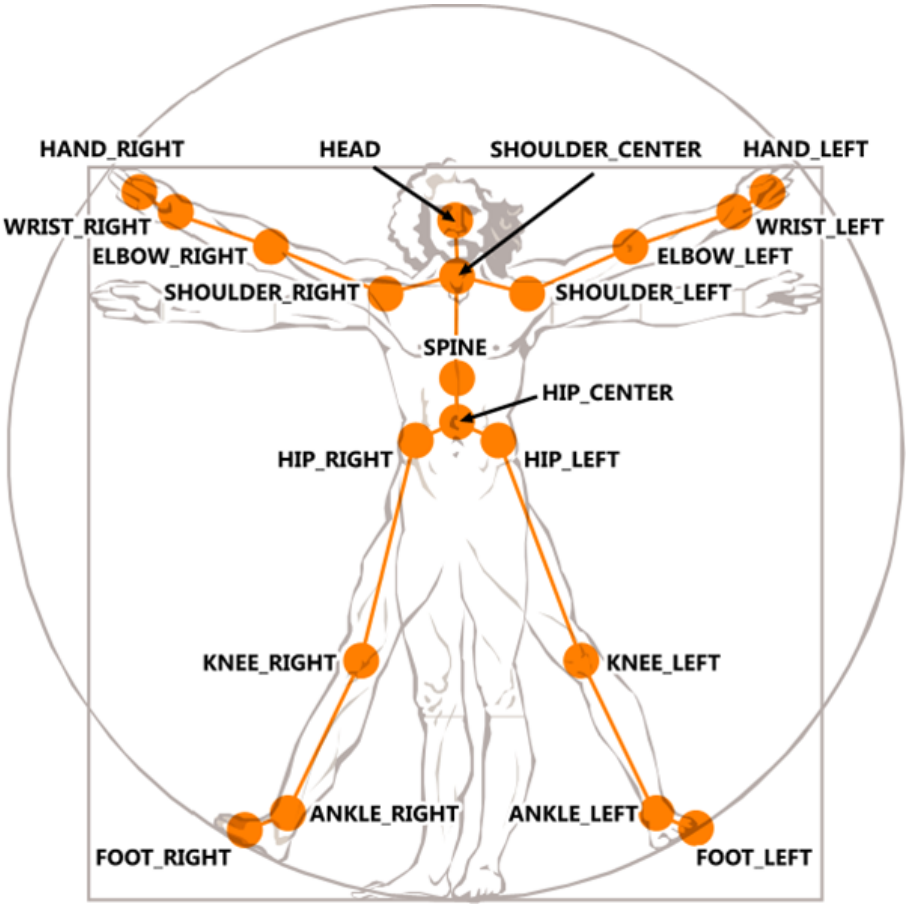
Abychom mohli tento stream plně využívat, musíme nejdříve povolit použití streamů `ColorImageStream`, `DepthImageStream` a `SkeletonStream`. Jako každý stream, má i tento událost, která je vyvolána, jakmile je snímek připraven. Jelikož tento stream není součástí objektu Kinect, musíme jej vytvořit. V konstruktoru tohoto objektu pak předáme referenci na Kinect. Při povolení streamu nastavíme formát barevného a hloubkového streamu.

Aby mohl tento stream zpracovat data, musíme mu je poskytnout. `BackgroundRemovedColorStream` obsahuje pro tento účel sadu metod. Pro zpracování barevných dat je zde metoda `ProcessColor`, která přijímá barevná data, pro hloubková data pak `ProcessDepth` analogicky přijímající hloubková data. Stejně tak je nutné poskytnout

data o kostře pomocí metody `ProcessSkeleton`. Ve všech případech se jedná o surová data z jednotlivých streamů. Nakonec je všem metodám předáno časové razítko daného snímku pro jasnou identifikaci. Jakmile jsou data zpracována, jsou poskytnuta v události `BackgroundRemovedFrameReady`.

InteractionStream

Abychom tento stream mohli využít, je nutné implementovat interface `IInteraction-Client`. Ten obsahuje pouze jedinou metodu a to `GetInteractionInfoAtLocation`. Implementace je velmi jednoduchá, stačí pouze vrátit objekt typu `InteractionInfo`. Ten obsahuje informace o pozici ruky, o jakou ruku se jedná a ke které kostře patří. Co je však nejdůležitější, dokáže nám tento stream vracet informace o tom, zdali je ruka otevřená či zavřená.



Obrázek 5: Kostra rozdělena na 20 jointů [11]

3 Edukin

Edukin vznikl roku 2012 jako projekt pro soutěž Imagine Cup[5]. Jedná se o komplexní ekosystém pro vývojáře nových aplikací a jednoduchých zábavných her. Ty jsou určeny jak hendikepovaným pacientům, tak i obyčejným uživatelům. Hlavním cílem projektu je rozvíjet hrubou a jemnou motoriku hendikepovaných pacientů pomocí zábavných her a aplikací. Jelikož projekt Edukin spolupracuje s ústavem pro tělesně postižené, byl pro rozvoj hrubé motoriky po poradě s odbornými pracovníky vybrán pohybový senzor Kinect. Velkým přínosem tohoto projektu je fakt, že infrastruktura uživatelského nastavení aplikací a samotného obsahu je uložena na cloudovém úložišti.

Hotová řešení ve specializovaných centrech sice plní své účely, avšak problém nastává v situaci, kdy je jedno zařízení sdíleno mezi více pacienty. Z finančních důvodů není tato situace vůbec neobvyklá. Jelikož aplikace nejsou koncipovány pro použití mezi více uživateli, vzniká problém konfliktu uživatelského obsahu. Pokud uživatel začne pracovat na jiném zařízení, jeho osobní data zde nejsou dostupná. Následná synchronizace je pak velmi obtížná, někdy zcela nemožná. Díky použití cloudového úložiště se může klient přihlásit pod svým účtem, tudíž aplikace pracuje pouze z jeho daty[6]. Systém byl tedy primárně navržen proto, aby ulehčil práci pracovníkům ve specializovaných centrech a zaručil pacientům při používání aplikací uživatelský komfort. Projekt se také zúčastnil soutěže Nápad roku 2012, kde obsadil 6 místo.

3.1 Oblasti projektu Edukin

Celý tento systém můžeme rozdělit do tří oblastí:

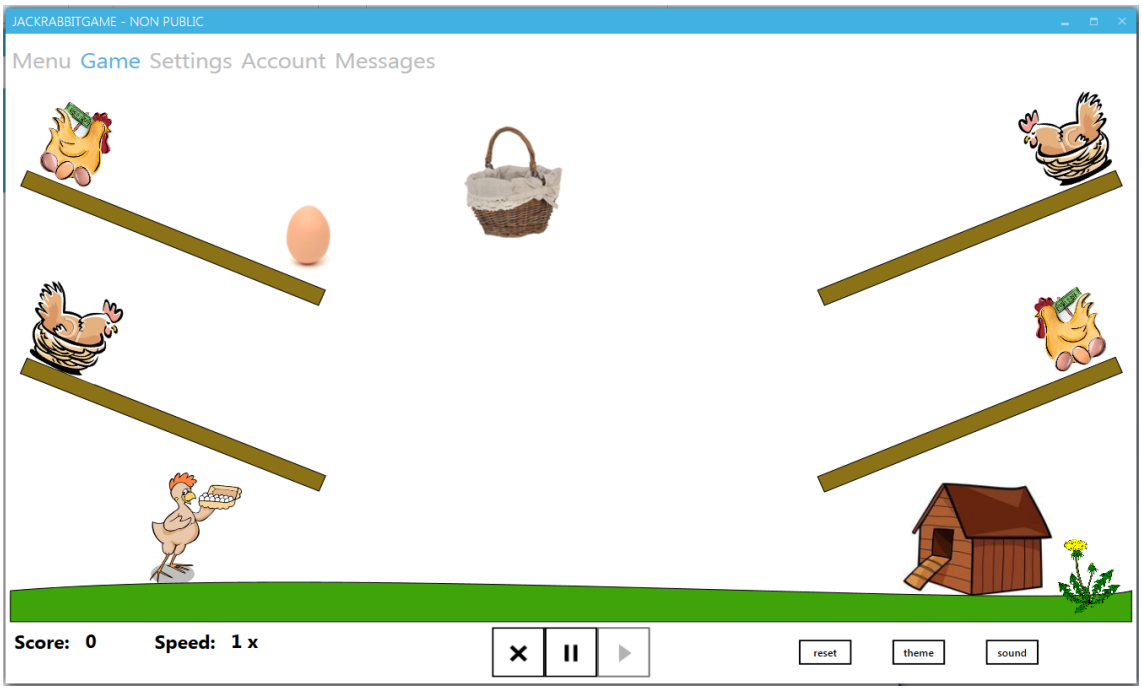
1. Dotyková zařízení - rozvoj jemné motoriky
2. Kinect - rozvoj hrubé motoriky
3. Cloudové prostředí - správa uživatelských účtů a nákupů

V celém projektu vzniklo několik frameworků, které usnadňují práci budoucím vývojářům aplikací. Frameworky jsou vytvořeny pro zařízení s operačními systémy Android a Windows.

Framework určený pro aplikace běžící pod operačním systémem Windows obsahuje celkem 7 částí. Chce-li vývojář použít k ovládání aplikace Kinect, je pro něj již připravená sada funkcí, které zprostředkovávají základní operace nad senzorem Kinect jako je inicializace a spuštění senzoru (*Edukin.Fw.Kinect*). Tato assembly se pak dále dělí na část pro práci se streamy (*Extensions*) a pro práci s gesty (*Gestures*). Dále framework obsahuje grafické komponenty, které mohou vývojáři použít pro jednotný vzhled (*Edukin.Fw.Resources*). Jelikož v systému figuruje několik rolí, existuje knihovna pro práci s uživateli a jejich autentizaci (*Edukin.Fw.Security*). Součástí frameworku jsou ještě rozšíření zahrnující práci se zvukem *Edukin.Fv.Audio* a běžně používanými funkcemi napříč celým systémem *Edukin.Fw.Common*.

3.2 Vytvořené aplikace v projektu Edukin

Pro dotyková zařízení s operačním systémem Android vznikla aplikace, kde je cílem správně asociovat tvary a obrázky. Pro senzor Kinect a operační systém Windows byly vyvinuty celkem tři aplikace. Jednou z implementovaných aplikací je hra s názvem Zajíc. Cílem v této hře je zachytávání padajících objektů, které jsou generovány na základě podnětů asistenta. Ty se zobrazují na hrací ploše a poté padají ze čtyř nakloněných rovin. Ruka uživatele je namapována na objekt, do kterého musí uživatel padající objekty chytat. Hra obsahuje 2 různá témata, tudíž je možné chytat vajíčka do košíku, nebo peníze do pokladničky. Po dosažení předem nastaveného skóre je hra ukončena. Mezi další implementované hry pro projektu Edukin patří hra Házení šipkami a Obtahování tvarů [5].



Obrázek 6: Ukázaka hry Zajíc [5]

3.3 Začlenění aplikace do projektu Edukin

Aplikace, která je výsledkem této práce, by měla být začleněna do systému Edukin. To zahrnuje využití již existující infrastruktury uživatelských účtů pro ukládání dosažených výsledků a nastavení aplikace do cloudového úložiště. Přihlášení do samotné aplikace bylo plánováno pomocí již hotových formulářů s využitím knihovny `Edukin.Fw.Security`, která je obsažena ve frameworku Edukin. Cloudové úložiště, na kterém byl projekt Edukin vyvíjen, je založeno na platformě Windows Azure, která navíc využívá SQL databázi.

Platforma Azure je zpoplatněná dle potřeby dané aplikace a to jak z hlediska výkonu, tak maximální velikosti databáze. Pro studenty je možné využít 30 denní zkušební verzi zdarma. Projektu Edukin se podařilo získat tuto platformu předplacenou na dobu dvou let a to díky umístění se v soutěži Microsoft Imagine Cup. Účty jsou přímo vázány na účastníky soutěže Adama Kašpara, Vojtěcha Bojka a Davida Martiníka, kteří již dokončili studium na Vysoké školy báňské - TU Ostrava. Po skončení dvouleté lhůty je plánována migrace na školní servery především z finančních důvodů. Pokud se systém dále bude používat, reimplementace cloudové infrastruktury bude nutná. Ta je relativně složitá a není využita tak, jak bylo původně plánováno. Celý systém obsahuje několik částí, které ale nejsou zcela dokončeny.

Aby bylo možné během vývoje nasimulovat prostředí, ve kterém vytvořené aplikace běží, bylo nutné získat informace o schématu databáze. Ty byly dostupné pouze z diplomových prací vývojářů a bohužel nebyly zcela kompletní. Ve schématech chyběly důležité informace o datových typech, integritních omezeních atp. Schéma bylo sice možné exportovat z databázového serveru, ten je však bohužel v ostrém provozu a obsahuje citlivá uživatelská data, kvůli kterým nebylo možné potřebná schémata získat.

Obrázek 7: Schéma platformy Edukin [5]

Hlavní programátoři projektu Edukin se již ve školním prostředí nepohybují a další vývoj byl pozastaven. Budoucnost tohoto projektu závisí na zájmu studentů či případném získání grantu. V tuto chvíli neexistuje žádná dokumentace k vytvořeným knihovnám a rozhraním. Informace o fungování tohoto projektu byly získávány převážně z diplomových prací a zdrojových kódů samotných tvůrců. Navržená infrastruktura je vázána na vlastníky účtů. Během vývoje došlo ke změně filosofie samotného projektu Edukin. Jeho infrastruktura je komplikovaná a není zcela hotová. Kvůli všem těmto aspektům bylo nutné vytvořit vlastní řešení.

Aby aplikace fungovala dle stanovených požadavků, byla pro ni po domluvě s vedoucím práce a panem Ing. Janem Martinovičem Ph.D. zřízena MS SQL databáze na školním serveru. Jelikož aplikace nespolupracuje s dalšími službami, není nutné, aby využívala složitou infrastrukturu jakou je Azure. Nově vytvořená infrastruktura založená čistě na SQL zabezpečuje všechny potřeby aplikace, které jsou vyžadovány. Mezi ně patří správa uživatelských účtů, vytvoření základních rolí a ukládání získaných dat.

4 Oblékání s Kinectem

Na začátku práce bylo nutné vymyslet koncept hry, která bude dostatečně zábavná a přitom dokáže jednotlivým pacientům sloužit k procvičení. Pro rozvoj hrubé a jemné motoriky jsme ve spolupráci s vedoucím práce navrhli hru s názvem Oblékání s Kinectem. V této kapitole bude vysvětleno, jak probíhá hraní samotné hry, co je jejím cílem a jaké jsou její možnosti. Důraz je hlavně kladen na intuitivní ovládání, jednoduchost a spolehlivost. I když je aplikace vyvíjena primárně pro ovládání pomocí pohybového senzoru Kinect, je možné ji kompletně ovládat i myší.

4.1 Popis hry

Cílem v této hře je obléct sebe sama do stejného oblečení, jako je vidět na vyobrazené předloze. Podle zvoleného téma jakým je například narozeninová oslava či sportovní událost je uživateli zobrazena předloha po celou dobu hry. Součástí uživatelského rozhraní je posunovací nabídka s dostupnými předměty. Z této nabídky může hráč vybírat jednotlivé části oblečení a ty umísťovat na příslušné místo. Celá předloha se skládá z pěti hlavních částí, které je potřeba nalézt a umístit na svou postavu. Těmi jsou klobouky, brýle, trička či košile, kalhoty a boty. Původní verze sice počítala s více objekty, jako jsou například doplňky, jenže ty jsou velmi malé a pro manipulaci pomocí Kinectu se díky své velikosti nehodí. Hra končí ve chvíli, kdy je právě oblékaná postava oblečena stejně, jako je vidět na vyobrazené předloze a na hrací ploše se nenacházejí žádné další objekty. Jelikož je hra určena primárně pro hendikepované osoby, obsahuje různé stupně nastavení, co se obtížnosti hry týče. Více v kapitole 4.4.

4.2 Uživatelské rozhraní

Hrací oblast se skládá ze tří hlavních ploch. Rozvržení ovládacích prvků na hrací ploše jsem konzultoval s vedoucím práce tak, aby byly snadno dosažitelné i pro hendikepované uživatele. Samotné ovládání není fyzicky náročné a k dosažení všech prvků na obrazovce stačí uživateli pohybovat rukou maximálně 50 centimetrů okolo těla. Aby byla aplikace zajímavá jak pro praváky, tak pro leváky, je možné celé uživatelské rozhraní otočit. Hru je možno ovládat jednou nebo dvěma rukama. Více o ovládání v sekci 4.4.3.

Předloha

V této oblasti je hráči zobrazena předloha, podle které se má obléci. Ta je buď vybrána z již dostupných témat, nebo je náhodně vygenerována jako jejich kombinace. Volbu oblékání podle pevně daného či náhodně vytvořeného téma je možné provést v nastavení před započetím hry. Hru je možné nastavit do volného režimu, v kterém není hráči zobrazena žádná předloha, hráč se tak může oblékat dle vlastního uvážení.

Aktuální stav

Pokud je při běhu aplikace dostupný Kinect, je v této části hrací plochy zobrazen výřez hlavy hráče z barevné kamery. V případě, že se nepodaří detekovat uživatele, je vyobrazen jednoduchý obličej a to podle pohlaví klienta zvoleného při registraci do systému. Cílem je do této oblasti dostat stejné oblečení, jaké je vidět na předloze.

Nabídka oblečení

Objekty, s kterými je možno manipulovat, jsou zařazeny pod sebou v rolovací nabídce. Tu je možné posunovat nahoru či dolů pomocí šipek na horní a dolní straně. Pokud je objekt z nabídky položen mimo hrací plochu, je do ní automaticky vrácen. V aktuální verzi se nachází tyto jednoduché typy oblečení, které ale pro účel hry dostačují. Hra je samozřejmě vytvořena tak, aby bylo možné jednotlivé obrázky kdykoliv vyměnit. Vytvoření jednotlivých typů oblečení je určitě prací pro zkušenějšího grafika.

Obrázek 8: Grafické rozhraní hry Oblékání s Kinectem

4.3 Průběh hry

Po spuštění aplikace je uživateli zobrazeno přihlašovací okno, v kterém je možné provést registraci či přihlášení viz kapitola 5.2. Nastavení hry je ukládáno do SQL databáze. Díky tomu se hráči po přihlášení do hry obnoví nastavení do stejného stavu, v jakém bylo před odhlášením.

Po přihlášení je hráči zobrazena úvodní strana s nastavením hry. Tlačítka jsou dostatečně velká a připravená na ovládání pomocí Kinectu. Ruka hráče je namapována do hry

jako virtuální ruka, která se pohybuje po hrací ploše. Uchopení předmětu je realizováno pomocí dvou různých gest a to v závislosti na tom, jaké gesto je aktuálně zvoleno, více v kapitole 4.4.3. Pokud se hráč nachází v sekci nastavení, je možné přepínat jednotlivá tlačítka pomocí gesta uchopení. Pokud se jedná o hráče se silným hendikepem, který by měl s nastavením problémy, může mu pomoci asistent, který změní nastavení pomocí myši. Po spuštění hry je úkolem hráče přesouvat na volnou plochu objekty z nabídky oblečení. Aby došlo k přesunu, je nutné provést gesto pro uchopení. Poté je předmět svázán s virtuální rukou a je možné jej přesunovat. Ve chvíli, kdy si hráč myslí, že daný předmět patří na dané místo, je nutné provést gesto upuštění. V případě umístění správného objektu je hráčovi započítán jeden kladný bod. Je-li předmět upuštěn na jiném místě, než na které patří, je připočten jeden záporný bod. Počet kladných a záporných bodů je vyobrazen v podobě usmívajících a mračících se smajlíků ve spodní části hry. Zpětná vazba pro pacienty je v tomto případě velmi důležitá. Každý úspěch či neúspěch je podtržen zvukovým doprovodem, aby bylo jasné, zda byla akce provedena správně či nikoliv. Aby bylo ovládání jednodušší, jsou oblasti pro umístění předmětů definovány jako čtverce o určitém rozměru. Pokud uživatel umístí správný objekt do tohoto čtverce a provede gesto upuštění, předmět je umístěn do správné pozice. Poté s ním již není možno pohybovat. Reakce aplikace v případě umístění špatného předmětu do hrací plochy záleží na konkrétním nastavení hry. V každém kole je zaznamenáván čas do úspěšného zreplikování dané předlohy. Abychom mohli kontrolovat výsledky hráče, je uloženo dosažené skóre a nastavení v jakém byla hra odehrána.

4.4 Možnosti nastavení

Nastavení je jedním z klíčových požadavků žádaných u aplikací tohoto typu. Důvodem je různý stupeň hendikepu jednotlivých klientů. Díky nastavení jsme schopni hru udělat zajímavou jak pro zdravého člověka, tak i pro člověka s hendikepem. Naprosto jednoduché činnosti a procesy pro zdravého člověka nemusí být už tak jednoduché a bezproblémové u hendikepovaných pacientů.

Vyběr strany

Ve spodní části se nachází tlačítko pro přepnutí strany. Po kliknutí se celý design hry horizontálně otočí. Tato volba je v aplikace hlavně z důvodu obrácené laterality některých hráčů. Díky tomu je nabídka blíže k dominantní ruce a hráč nemusí pro dosažení předmětů natahovat ruku. Díky častému zvedání ruky hráče je pravděpodobné, že se hráč unaví. Otočení má tedy určitě smysl, abychom tuto situaci co nejvíce oddálili.

4.4.1 Mód hry

Ve hře jsou dostupné celkem tři módy. Ty mění způsob, jakým jsou generovány předlohy.

Bez předlohy

V tomto režimu není hráči zobrazena žádná předloha. Hráč se může obléci dle jeho vlastní vůle. Tento mód je vhodný právě pro pacienty, kteří mají problém s asociováním různých částí oblečení. Pacient se v tomto módu může zaměřit pouze na rozvoj hrubé motoriky a nemusí se zabývat tím, který předmět má do hrací plochy umístit. Hra v tomto modu končí ve chvíli, když je postava kompletně oblečená a na hrací ploše se nenacházejí žádné další volně upuštěné objekty.

Dle předlohy

V módu s předlohou je hráči zobrazena postava oblečená dle zvoleného motivu, jako je například narozeninová oslava či sportovní činnost. Jelikož jsou objekty patřící do jedné sady podobně stylizovány, ulehčuje to hráči výběr. Hra končí stejně, jako tomu bylo v minulém případě.

Náhodný dle předlohy

Při výběru náhodného módu dle předlohy je z databáze oblečení a doplňků náhodně vybráno několik objektů, které se zobrazí jako předloha. Hráč se pak musí obléci podle zvolené předlohy. Tento mód byl vytvořen z jednoho prostého důvodu. Tím je osvěžení samotné hry, jelikož typy oblečení pro jednotlivá témata jsou jednoduše zapamatovatelná. Konec hry nastává stejně jako v předchozích případech.

4.4.2 Téma

Pro hráče je připravena škála motivů, které může použít jako předlohu. Tato témata lze vybírat jen v módu s předlohou. Mezi základní nabízená témata patří nakupování, narozeniny, tanec a sport.

4.4.3 Nastavení složitosti hry

Nápověda vysvícením objektu

Zapnutím této volby se aktivuje nápověda na pozadí. Ta monitoruje pohyb rolovací nabídky. Pokud je ve viditelné části zobrazen objekt, který patří na hrací plochu, je na něj aplikována animace, která jej rozbliká, čímž napoví uživateli, aby jej umístil na patřičné místo na své postavě. Časový limit pro rozblikání je nastaven 10 vteřin od zastavení rolovací nabídky.

Mizení objektů při nesprávném umístění

Pokud je tato volba zapnuta a hráč nesprávně umístí objekt na hrací plochu, je z ní automaticky odebrán. Jelikož hráč nemusí vracet špatně umístěný předmět zpět do nabídky, je toto nastavení vhodné především pro začátečníky či hendikepované pacienty s horší fyzickou výdrží.

Kategorizace

Díky kategorizaci můžeme hendikepovaným zjednodušit výběr z nabídky předmětů. Pro některé hráče nemusí být nalezení správné části vůbec snadný úkol, díky zhoršené schopnosti rozeznávat některé detaily. Pokud seřadíme stejné typy položek pod sebe, je do určité míry zmenšena náročnost a hra může být více zábavná.

Ovládání

Ovládat aplikaci může hráč dvěma způsoby.

- **Ovládání jednou rukou**

  V tomto případě je uchopení předmětu realizováno sevřením a upuštění rozevřením hráčovy dlaně. Hra je vytvořena tak, aby jako virtuální ukazatel sloužila vždy ta ruka, která je výše. I když je možné ovládat hru oběma rukama současně, je přesun, uchopení a upuštění realizováno jen jednou rukou.

- **Ovládání dvěma rukama**

  U této aplikace byl kladen požadavek na zapojení obou rukou hned z několika důvodů. Jedním z nich je procvičení obou horních končetin pacienta. Dalším důvodem je to, že pro určitou skupinu pacientů není jednoduché jasně realizovat gesto pouze pomocí jedné ruky. Kvůli tomu je možné přepnout způsob manipulace s předměty na obě ruce. Pohyb po ploše je namapován na pravou ruku a gesto uchopení je simulováno umístěním levé ruky do polohy před břicho hráče. Položení objektu pak nastává v situaci, kdy je levá ruka vrácena do klidové polohy vedle těla.

5 Implementace aplikace

Hra je vytvořena jako projekt ve Windows Presentation Foundation v programovacím jazyce C#. Jelikož hra obsahuje velké množství objektů a grafických prvků, s kterými je manipulováno, není formulářová struktura Windows Forms vhodná.

O ukládání samotných dat se stará Microsoft SQL server ve verzi 2012, pro lokální databázi je použitý SQL Compact Edition 4.0. Jelikož vnitřní logika aplikace není vázána na projekt Edukin, nebylo využito již vytvořeného frameworku a aplikace je postavena úplně od základu. Framework již obsahuje sadu vytvořených gest, ty však nevyhovují požadavkům této aplikace. Pro ovládání aplikace byla vytvořena nová gesta. Více v kapitole 5.10.1.

5.1 Technické specifikace

Aby bylo možné aplikaci bezproblémově používat, je nutné splnit minimální softwarové a hardwarové požadavky. Samotná hra není nijak hardwarově náročná, avšak pro plynulé hraní je potřeba splnění minimálních požadavků, které jsou zmíněny v kapitole Kinect 2.3. Samozřejmostí je internetové připojení, které je nutné pro registraci nového uživatelského účtu a dále pak pro ukládání uživatelských dat. Po úspěšné registraci je možné hrát i bez internetového připojení. Mezi softwarovou výbavu pak patří .NET framework ve verzi 4.5.1 a ovladače ke Kinectu. Důležitým faktem zůstává, že aplikace je schopná běžet v prostředí bez vývojářských nástrojů, ale pouze na zařízení Kinect pro Windows. Toto omezení je ze strany Microsoftu a musíme jej respektovat. Pokud chceme aplikaci používat i se zařízením Kinect pro Xbox, musíme ještě dodatečně nainstalovat 600 MB balíček Kinect SDK.

5.2 Dostupnost aplikace

Jelikož aplikace ukládá výsledky a nastavení pro jednotlivé uživatele, je nutné, aby měl každý uživatel svůj vlastní účet. Vytvoření účtu probíhá přímo v aplikaci. K dokončení registrace stačí zadat pouze několik nutných údajů. To lze provést přímo v samotné aplikaci hned při zapnutí, kliknutím na tlačítko registrace. Pro aplikaci není dostupné žádné webové rozhraní, přes které by se mohl uživatel registrovat, registrace probíhá přímo v aplikaci.

Aplikace je součástí projektu Edukin a pro správu uživatelských dat je použit školní server s Microsoft SQL databází. Aplikace používá ještě lokální databázi. Její využití má své opodstatnění, jelikož se v době používání aplikace může stát, že internetové připojení nebude dostupné. Pokud tato situace nastane, všechny výsledky, včetně nastavení uživatele jsou ukládány do lokální databáze. Při dalším spuštění aplikace jsou pak nastavení s výsledky uživatele synchronizovány do vzdálené databáze. Abychom mohli zaznamenávat dosažené výsledky a ukládat uživatelská data včetně nastavení pro konkrétního uživatele i v době bez internetového připojení, je nutné, aby se uživatel na daném počítači alespoň jednou úspěšně přihlásil vůči vzdálené databázi. Pokud se po přihlášení uživa-

tele účet nenachází v lokální databázi, je do ní automaticky synchronizován pro budoucí použití v offline modu.

Uživatelské rozhraní bylo vytvořeno pomocí jazyku XAML. Díky podpoře škálování u všech prvků jsme schopni nastavovat různou velikost grafického rozhraní v závislosti na rozlišení. Po přihlášení je aplikace přepnuta do celoobrazovkového režimu. Minimální rozlišení pro aplikaci není omezeno, pro dostatečný komfort je však doporučeno minimílně HD rožlišení. Aplikace byla testována na dvou různých rozlišeních a to 1366*768 a 1920*1080.

5.3 Role a scénáře

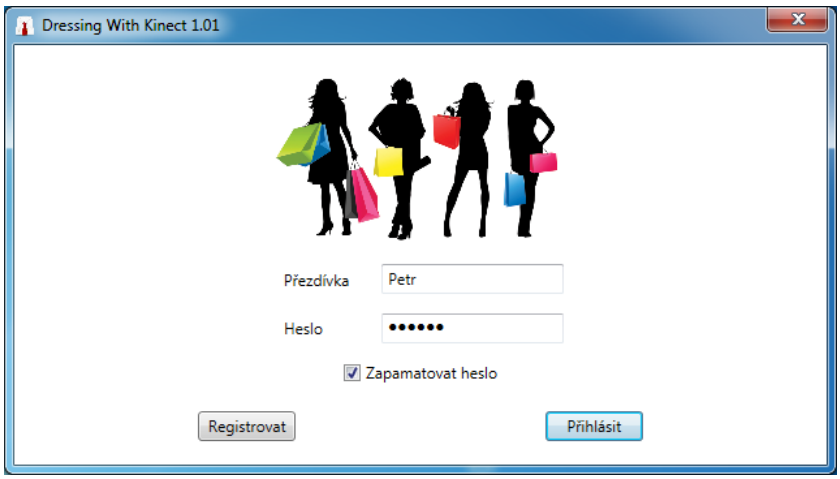
Aby byla aplikace použitelná v prostředí ústavů, denních stacionářů a jiných zařízení existují v ní dvě hlavní role. Těmi jsou učitel a hráč. Každá role má jiná práva a podle nich se uživatel v aplikaci pohybuje. Role je určena při registraci uživatele do systému.

Učitel

Uživatel v této roli je schopen zaregistrovat nové hráče pod svůj vlastní účet. Tato role je v aplikaci z důvodu usnadnění přihlašovacího procesu hendikepovaným hráčům. Takto vytvoření hráči poté nepotřebují pro přihlášení do aplikace zadávat heslo. Učiteli je v aplikaci zobrazen seznam a základní informace o jeho hráčích.

Hráč

Do této role patří automaticky každý uživatel, kromě učitele, který je do aplikace registrován. Pokud je hráči přiřazen učitel, platí pro něj jiná bezpečnostní pravidla, než pro klasického hráče. Tím se myslí povinnost zadávat heslo při přihlášení do aplikace. To má své opodstatnění a tím je jednodušší přístup k aplikaci pro klienty jednotlivých zařízení.



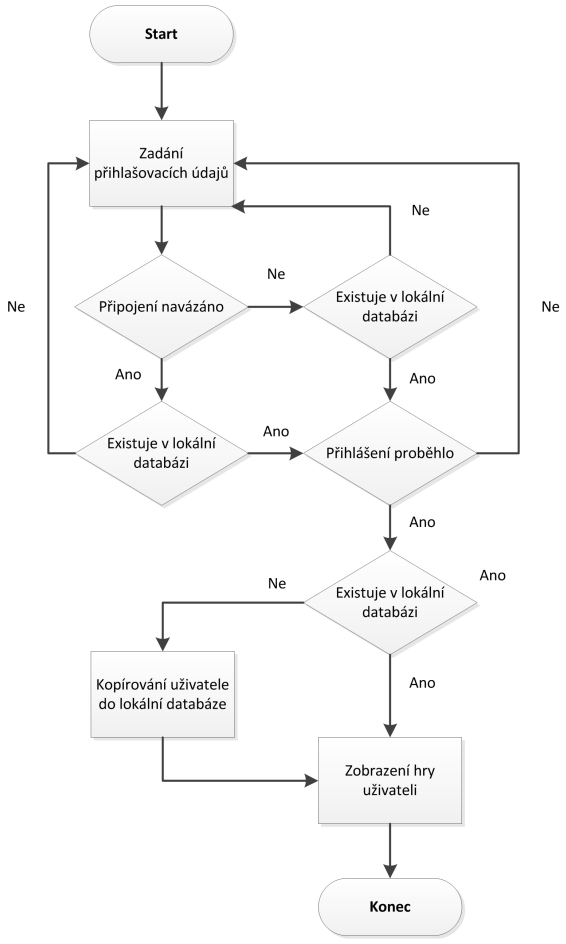
Obrázek 9: Přihlašovací obrazovka aplikace

5.4 Diagramy aktivit

Popis jednotlivých procesů můžeme popsat pomocí diagramu aktivit. Díky tomu lépe porozumíme samotnému fungování jednotlivých částí aplikace.

Přihlášení uživatele

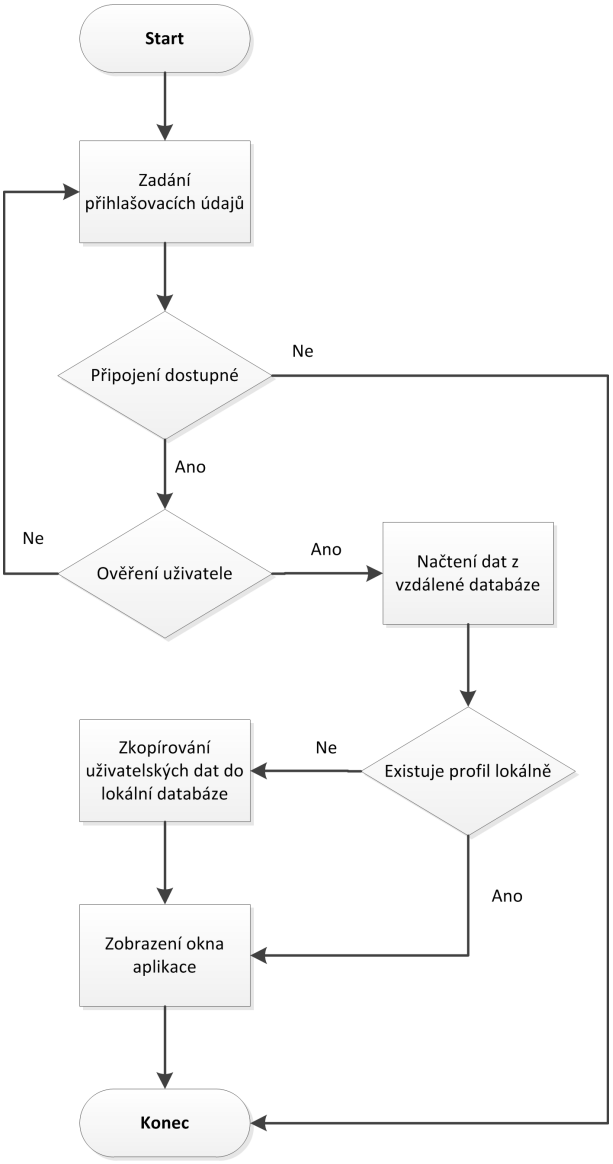
Před samotným přihlášením se nejprve otestuje internetová konektivita, respektive se aplikace pokusí navázat spojení se vzdálenou databází. Pokud je výsledkem této akce úspěch, přihlášení probíhá vůči vzdálené databázi. Aplikace prověří, zda je profil hráče již uložen lokálně, aby bylo možné tohoto uživatele přihlásit i v době, kdy není zaručena internetová konektivita. Pokud není profil hráče nalezen, je do lokální databáze zkopírován včetně jeho nastavení. V případě, že se spojení nepodaří navázat, pokusí se aplikace vyhledat uživatele v lokální databázi. Je-li nalezen, přihlášení je provedeno pouze lokálně. Výsledky jsou poté také ukládány lokálně.



Obrázek 10: Diagram aktivity přihlášení uživatele

Vytvoření kopie v lokální databázi

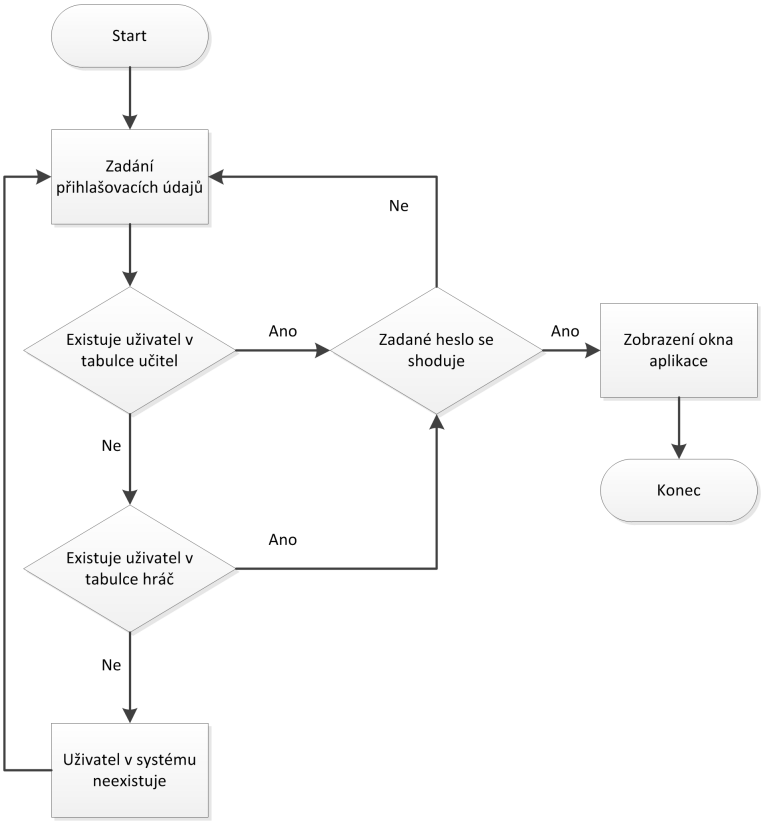
Podaří-li se uživateli přihlásit vůči vzdálené databázi, aplikace ověří, zda již tento profil existuje lokálně. V případě, že se data o přihlášeném uživateli v lokální databázi nevyskytují, je jeho profil zkopírován pro pozdější použití offline.



Obrázek 11: Diagram aktivity zkopírování uživatele

Udělení role

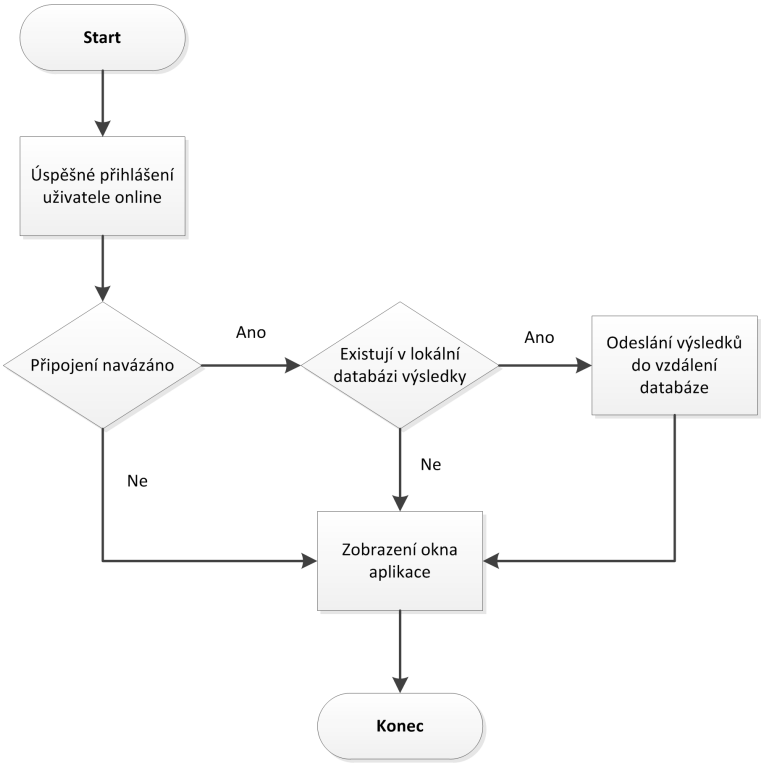
Role je realizována pomocí dotazu na existenci záznamu v dané tabulce. Nachází-li se zadaná přezdívka uživatele v databázi a hesla se shodují, je uživateli udělena konkrétní role. Pokud nebude dostupné připojení ke vzdálené databázi, je použit stejný postup, pouze s tím rozdílem, že jsou uživatelé hledáni lokálně.



Obrázek 12: Diagram aktivity udělení role

Synchronizace výsledků

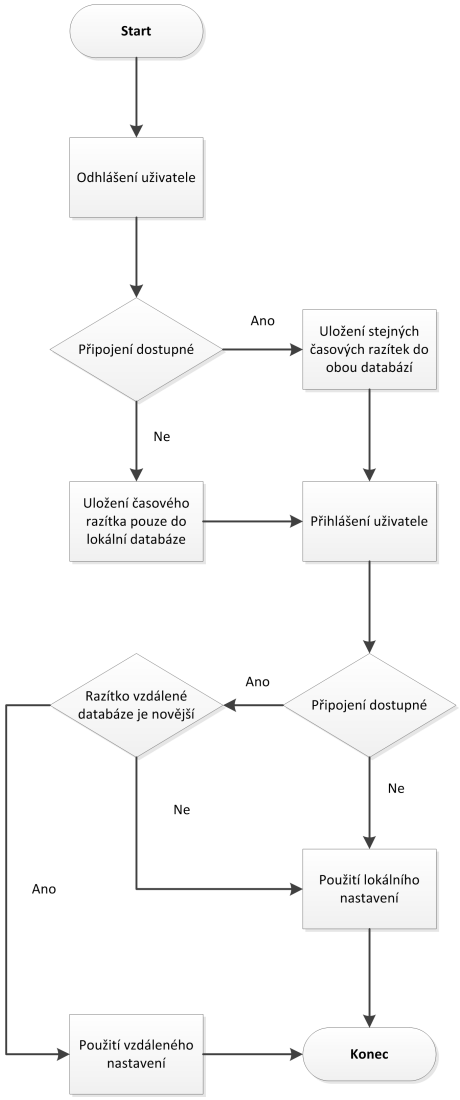
V případě, že se uživatel přihlásí do systému, zkontrolujeme, zda existují nějaké lokální výsledky. Pokud ano a je dostupné internetové přpojení, jsou tyto výsledky synchronizovány do vzdálené databáze. Z lokální databáze jsou poté smazány.



Obrázek 13: Diagram aktivity synchronizace výsledků

Synchronizace nastavení

Nastavení pro daného hráče se ukládá před samotným odhlášením hráče z aplikace. Jelikož chceme hráči poskytnout komfort, a to i v době bez internetového připojení, musíme zaručit, že nastavení aplikace zůstane stejné, jako tomu bylo před vypnutím. Proto je nastavení uživatele přiřazeno časové razítko, které určuje, které nastavení je novější. V případě, že je hráč online, jsou uložena stejná časová razítka do obou databází. V době bez internetového připojení je časové razítko uloženo jen do lokální databáze. Pokud je připojení dostupné, tak při dalším přihlášení uživatele zkontrolujeme časová razítka z obou databází a použijeme to, které je novější.



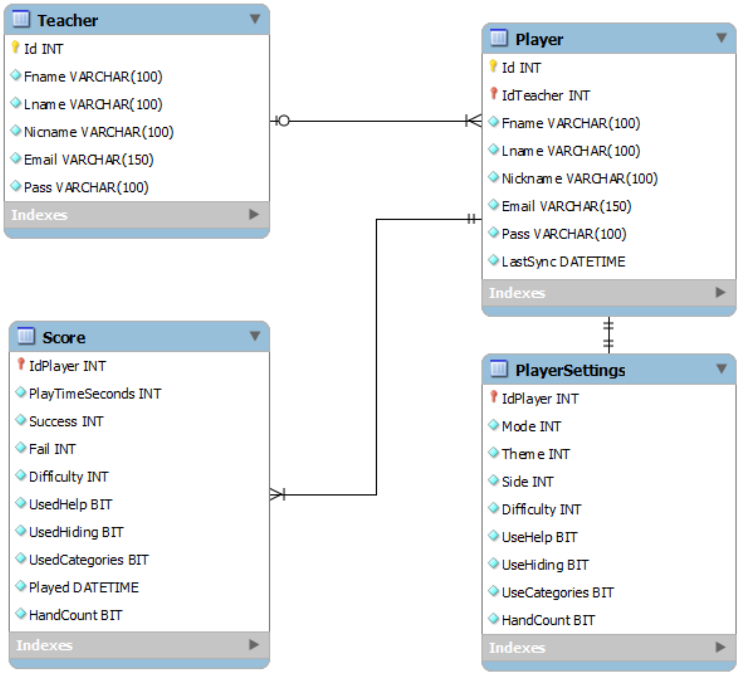
Obrázek 14: Diagram aktivity synchronizace nastavení

5.5 Databázová analýza

Následující analýza by nám měla prozradit, s jakými daty a entitními typy aplikace pracuje. Schéma obou databází je stejné. Liší se jen v minimálních detailech, které budou popsány níže.

5.5.1 ER diagram

ER diagram je jedním z jednoduchých způsobů, kterým můžeme popsat databázové schéma. Schéma této databáze není vůbec složité, což je také zapříčiněno tím, že aplikace funguje samostatně bez napojení na další části Edukinu, jak již bylo řečeno v kapitle 3.



Obrázek 15: ER diagram databáze

Bezpečnost

Aby bylo zamezeno útokům typu SQL Injection, jsou všechna data vkládaná do formulářů validována a to pomocí regulárních výrazů. Databázová vrstva, která se stará o vytváření SQL dotazů používá výhradně skládání pomocí parametrů. Tento způsob automaticky odstraňuje všechny nepovolené znaky typu uvozovka, komentář a další, které jsou typické pro útoky pomocí SQL Injection. Heslo uživatele samozřejmě není uloženo ve formě čistého textu, ale před uložením je aplikována hashovací algoritmus BCrypt. Ověření uživatele pak probíhá výpočtem hashe ze zadaného hesla a následným porovnání se

záznamem v databázi. Bezpečnost je zaručena díky relativně velké výpočetní složitosti tohoto hashe v porovnání se známými hashovacími funkcemi jakými jsou MD5 či SHA-1. Díky tomu by měla být uživatelská hesla v bezpečí a to i v případě kompromitace databáze [12].

Odlišnosti databází

Pro vytvoření uživatelského identifikátoru se využívá integrované funkce MS SQL serveru známého jako identity. Tato funkce přiřadí každému dalšímu záznamu unikátní identifikátor, díky kterému jsme schopní záznam zpětně dohledat. Tato funkce je použita ve vzdálené databázi u tabulky `Teacher` a `Player`. V lokální databázi však tuto funkcionalitu nepotřebujeme, jelikož slouží pouze jako dočasná kopie dat hráče a jeho nastavení.

5.6 Lineární schéma entit

Název	(primární klíč, *cizí klíč*, atributy)
Teacher	(Id, Fname, Lname,Nickname, Gender Email, Pass)
Player	(Id, *IdTeacher*, Fname, Lname,Nickname, Gender Email, Pass, LastSync)
Score	(*IdPlayer*, PlayTimeSeconds, Success, Fail, Difficulty, UsedHelp, UsedHiding, UsedCategories, Played, HandCount)
PlayerSettings	(*IdPlayer*, Mode, Theme, Side, Difficulty, UseHelp, UseHiding, UseCategories, HandCount)

5.7 Lineární schéma závislostí

Název	(rodičovská entita, dětská entita)
FKTeacherPlayer	(Teacher, Player)
FKPlayerPlayerSettings	(Player, PlayerSettings)
FKPlayerScore	(Player, Score)

5.8 Datový slovník

V tabulkách níže se můžeme podívat na jednotlivé entitní typy, a jak jsou reprezentovány v databázích. Ve sloupci popis je vysvětleno, co daný atribut znamená. Integritní omezení nám říkají, v jakém formátu jsou data ukládána.

Název sloupce	Datový typ	Null	Popis	IO
Id	INT	NE	Id záznamu	Unikátní
Fname	NVARCHAR(100)	NE	Jméno	1 - 100 znaků
Lname	NVARCHAR(100)	NE	Příjmení	1 - 100 znaků
Nickname	NVARCHAR(100)	NE	Přezdívka, primární klíč	1 - 100 znaků
Gender	BIT	NE	Pohlaví	-
Email	NVARCHAR(150)	NE	Email	Validní, unikátní
Pass	NVARCHAR(150)	NE	Heslo	1 - 150 znaků

Tabulka 2: Entita Teacher

Název sloupce	Datový typ	Null	Popis	IO
Id	INT	NE	Id záznamu	Unikátní
IdTeacher	INT	ANO	Cizí klíč na učitele	-
Fname	NVARCHAR(100)	NE	Jméno	1 - 100 znaků
Lname	NVARCHAR(100)	NE	Příjmení	1 - 100 znaků
Nickname	NVARCHAR(100)	NE	Přezdívka, primární klíč	1 - 100 znaků
Gender	BIT	NE	Pohlaví	-
Email	NVARCHAR(150)	NE	Email	Validní, unikátní
Pass	NVARCHAR(150)	ANO	Heslo	1 - 150 znaků
LastSync	DateTime	NE	Čas poslední synchronizace	-

Tabulka 3: Entita Player

Název sloupce	Datový typ	Null	Popis	IO
IdPlayer	INT	NE	Id uživatele	-
PlayTimeSeconds	INT	NE	Délka hry	-
Success	INT	NE	Počet platných pokusů	-
Fail	INT	NE	Počet neplatných pokusů	-
Difficulty	INT	NE	Obtížnost hry	-
UsedHelp	BIT	NE	Použita nápověda	-
UsedHiding	BIT	NE	Zapnuto mizení předmětů	-
UsedCategories	BIT	NE	Třídění do kategorií	-
Played	DateTime	NE	Čas a datum hry	-
HandCount	BIT	NE	Počet rukou použitých k ovládání	-

Tabulka 4: Entita Score

Název sloupce	Datový typ	Null	Popis	IO
IdPlayer	INT	NE	Id uživatele	-
Mode	INT	NE	Délka hry	-
Theme	INT	NE	Počet platných pokusů	-
Side	INT	NE	Počet neplatných pokusů	-
Difficulty	INT	NE	Obtížnost hry	-
UseHelp	BIT	NE	Nápověda zapnuta	-
UseHiding	BIT	NE	Mizení předmětů zapnuto	-
UseCategories	BIT	NE	Třídění zapnuto	-
HandCount	BIT	NE	Počet použitých rukou	-

Tabulka 5: Entita PlayerSettings

5.9 Prezentační vrstva

Tato vrstva je zodpovědná za grafické zobrazení a vizualizaci všech procesů probíhajících v aplikaci. Navigace je realizována pomocí přechodu mezi jednotlivými stránkami, podobně jako je tomu u webových aplikací. Hlavní aplikace je tvořena oknem, v kterém se nachází komponenta `Frame`, která slouží jako rodičovský kontejner pro jednotlivé stránky. Ty jsou tvořeny pomocí komponenty `Page`. Tento způsob navigace byl zvolen z důvodu jednoduché správy a dobré segmentace jednotlivých částí prezentační vrstvy. Mohli bychom sice použít komponenty typu `Window`, ale díky použití `Frame` jsme schopní přecházet mezi jednotlivými stránkami bez zavírání a otevírání oken, což je určitě uživatelsky přívětivější.

Grafické rozhraní je složeno celkem z těchto stránek:

- Login.xaml - přihlašovací formulář
- Register.xaml - registrační formulář
- TeacherManage.xaml - registrace a správa uživatelů učitele
- Game.xaml - samotná hra

Všechny takto vytvořené stránky se nacházejí v adresáři `DressingWithKinect.Core.Pages`. Hlavní komponenta `Game` je složena z mnoha dalších komponent, které jsou součástí WPF jako je `Canvas`, `Grid` nebo `ViewBox`. Mezi vytvořené komponenty patří `ClothRotator`, který představuje nabídku předmětů pro uživatele. Abychom mohli komponentu flexibilně využívat, můžeme měnit počet položek v nabídce. Metoda `AddPlaceholders` se poté již postará o vytvoření požadovaného počtu buněk. Komponenta obsahuje metodu `Scroll`, která je zodpovědná za pohyb nabídky oblečení nahoru a dolů. Jelikož potřebujeme vědět, kdy se pohyb nabídky zastavil, je v rámci třídy dostupná událost `ScrollChanged`, která je vyvolána vždy, když je zaznamenán pohyb

nabídky. Od této doby je pak měřen časový interval. Jakmile je překročena předem definovaná doba a v nabídce se nachází alespoň jeden objekt, který patří na hráčovu postavu, je na tento předmět aplikována animace.

V rámci aplikace bylo vytvořeno ještě několik dalších komponent pro pohodlné ovládání pomocí Kinectu. Mezi ně se řadí komponenta pro změnu nastavení aplikace `SettingsLayout`. Ta v sobě obsahuje další tři hlavní komponenty `ThemeSelector`, `ModeSelector` a `Options`. Jak již názvy napovídají, `SettingsLayout` je pouze rodičovská komponenta pro všechny tři ostatní a určuje jejich rozložení. Tyto komponenty obsahují obrázky, které slouží jako tlačítka. Aby bylo patrné, které volby jsou aktuálně vybrány, je přes obrázky překreslen zelený symbol zaškrtnutí. V případě, že hra skončí je uživateli zobrazeno okno s dotazem, jestli chce hru opakovat ve stejném módu. Pro tuto situaci je také vytvořena komponenta s názvem `YesOrNoDialog`. Na bočních stranách aplikace se nachází komponenta `HelpInfo`, která uživateli zobrazuje, jaký je význam jednotlivých tlačítek.

Pokud momentálně aplikace sleduje hráče, je zobrazena virtuální ruka v hrací ploše. V případě, že je pěst hráče sevřená, ruka svítí zeleně. Otevřená pěst je pak indikována šedou barvou. Aby nedocházelo k frustrujícím situacím, kdy je předmět nechtěně upuštěn zcela mimo hrací plochu, je v aplikaci vytvořeno neviditelné ohraničení, za které není možné předmět umístit.

5.10 Aplikační vrstva

V této podsekci se dovíme něco o samotném fungování vnitřní logiky aplikace. Níže budou popsány metody pomocí kterých jsme schopni mapovat ruku hráče do hry, určovat jaké gesto hráč provádí či zobrazovat hlavu hráče na hrací ploše.

Logiku hry můžeme rozdělit do několika částí:

- Core - obsluha vstupů z myši a Kinectu
- Controls - ovládací prvky a komponenty
- GameModes - herní módy
- Utils - pomocné třídy a nápověda
- Pages - uživatelské rozhraní
- Network - databázová vrstva
- Objects - vlastní definované objekty používané napříč aplikací

Core

Jak již víme z předchozích kapitol, hru je možné ovládat pomocí myši nebo Kinectu. Zodpovědnost za zpracování podnětů z obou vstupních zařízení mají třídy `Engine`

a EngineKinect, obsažené v tomto adresáři. Obě tyto třídy jsou potomkem rodičovské třídy Common. Ta obsahuje sadu metod pro manipulaci s objekty, které jsou umístěny na hrací ploše. Vytvoření potomci této třídy mají přístup ke stejným datům, což je velkou výhodou. Na základě odlišných podnětů z obou periferií jsme schopni volat stejné metody.

Třída Engine je zodpovědná za zpracování požadavků, které jsou vyvolány myší. Jedná se o události MouseUp, MouseDown a MouseMove. Při události MouseDown, je objekt odebrán z aktuálního rodičovského kontejneru a je přiřazen do kontejneru Canvas, abychom s ním mohli pohybovat. MouseMove poté zprostředkovává pohyb samotného objektu po kreslícím plátně. Informace o tom, kde se kurzor myš nachází jsou dostupné přímo v argumentu události, tudíž jsme schopni podle nich nastavovat pozici objektu. Při upuštění je vyvolána událost MouseUp, v které se testuje místo položení. Rozšíření, které nám umožňuje jednotlivé elementy přesouvat je popsáno níže v kategorii Utils 5.10.

```
private KinectSensorChooser = new KinectSensorChooser();
private KinectSensor Kinect;

private void GetKinect()
{
    \\Chceme reagovat na všechny změny zařízení
    KinectSensorChooser.KinectChanged += SensorChooserOnKinectChanged;
    SensorChooserUi.KinectSensorChooser = KinectSensorChooser;

    \\Spustíme inicializaci
    KinectSensorChooser.Start();
}

private void SensorChooserOnKinectChanged(object sender, KinectChangedEventArgs args)
{
    if (args.NewSensor != null)
    {
        // Zapneme potřebné streamy a nastavíme jejich parametry

        // Získáme referenci na Kinect
        Kinect = args.NewSensor;
        if (Kinect.Status == KinectStatus.Connected)
            Kinect.Start();
    }
}
```

Výpis 1: Získání reference na Kinect pomocí komponenty Sensor Chooser

Třída EngineKinect obsahuje všechny metody potřebné pro práci s Kinectem. Pokud se Kinect podaří inicializovat, informuje nás o této události komponenta KinectSensorChooser. Získání reference je velmi jednoduché, jelikož má komponenta vnitřní vlastnost Kinect, do které připojené zařízení přiřadí. V události SensorChooserOnKinectChanged nastavíme parametry s kterými bude Kinect spuštěn a pokud vše proběhne bez problémů, zapneme Kinect.

Jelikož budeme v aplikaci pracovat s informacemi ze všech streamů, přihlásíme se k jejich odběru. To je možné pomocí události AllFramesReady. Ta je vyvolána vždy,

když jsou připraveny snímky ze všech tří streamů. Z argumentu událostí si poté získáme potřebná data, která s kterými budeme dále pracovat. Aplikace používá taktéž rozšiřující streamy `InteractionStream` a `BackgroundRemovedColorStream`.

Jednou z dalších metod obsažených v této třídě je samotná inicializace jak můžeme vidět výše. Velmi důležité jsou metody `KinectGrab` a `KinectRelease`. Ty jsou volány na základě informací o stavu ruky. Metoda `KinectGrab` se stará o uchopení předmětu, jeho svázání s virtuální rukou a přesun. Pokud je ruka stále sevřená, metoda nastavuje novou polohu objektu dle aktuální snímkovací frekvence. V případě, že je provedeno gesto pro upuštění je zavolána metoda `KinectRelease` a předmět je umístěn na hrací plochu. Metoda vrací pravdivostní hodnotu, která určuje, zda byl objekt správně umístěn. Další neméně důležitou metodou je `ProcessInteractionData`, která zpracovává informace z interaction streamu. V závislosti na tom, jaká data jsou v této metodě vyhodnocena, jsou volány výše zmíněné metody.

Abychom mohli zobrazit hlavu hráče, potřebujeme barevná data. Ta nám poskytne `ColorImageStream`. Tato data předáme instanci třídy `BackgroundRemovedColor-Stream`, která dokáže oddělit pixely, které patří hráči od okolí. Všechny pixely, které nepatří hráči mají nastavenu průhlednou složku na maximální hodnotu. Protože chceme zobrazit pouze hlavu hráče, musíme si nejprve zjistit, kde se v obraze hlava nachází. Toho docílíme pomocí informací ze skeleton streamu, přesněji zjistíme X-ovou a Y-ovou souřadnici z kostry, za pomocí jointu `Head`. Jelikož jsou souřadnice vázány na `SkeletonStream`, musíme je přemapovat na `ColorStream`. K tomu je určena zabudovaná třída s názvem `CoordinateMapper`. Ta obsahuje sadu metod pro převody mezi všemi formáty obrazových a skeletálních dat. Z těchto souřadnic jsme již schopni přesně určit, kde se hlava nachází.

```
private Point GetJointPoint(KinectSensor kinect, Joint joint)
{
    //Vytvoříme instanci, obsahující metody pro mapování bodů
    CoordinateMapper mapper = new CoordinateMapper(kinect);

    //Ekvivlentní bod hloubkové a barevné mapy získáme převodem souřadnic
    ColorImagePoint point = mapper.MapSkeletonPointToColorPoint(joint.Position, kinect.
        ColorStream.Format);
    return new Point(point.X, point.Y);
}
```

Výpis 2: Mapováni bodu z kostry na bod v barevném obraze

Aby byla zobrazena pouze hlava, musíme z obrazu odstranit zbytek těla hráče. K tomu slouží metoda `CropColor`, která přebírá jako jeden z parametrů barevný snímek, na kterém je v tomto případě tělo hráče. Výstupem této metody je opět barevný snímek, v němž se nachází pouze výřez hráčovy hlavy. Ten je umístěn na připravené místo v předloze, stejně jako na hrací ploše. V případě, kdy před Kinectem nestojí žádný hráč, je stream z barevné kamery nahrazen statickým obrázkem podle pohlaví hráče.

Objects

Pro jednodušší práci s částmi oblečení je vytvořena třída s názvem `ClothItem`, která tvoří abstrakci nad samotným obrázkem. Tato třída v sobě obsahuje informace, o jakou část oblečení se jedná, do kterého téma patří, je-li umístěna na správném místě, jaká je jeho aktuální poloha na plátně a podobně. Ze stejného důvodu je také v aplikaci vytvořena třída `PlaceHolder`, kterou můžeme použít pro definování předloh.

GameModes

V tomto adresáři jsou vytvořeny tři třídy pro reprezentaci herního módu, který je právě spuštěn. Všechny tyto třídy využívají statickou třídu `Mode`, která obsahuje metody, které jsou pro všechny módy společné. Příkladem může být metoda `SortByParts`, která seřadí objekty v nabídce podle daného typu. Opakem je metoda `Randomize`, která předměty náhodně prohází mezi sebou. Důležitá je metoda `SelectClothForRotator`, která vytváří obsah samotné nabídky, v závislosti na nastaveném módu.

Utils

Součástí tohoto adresáře jsou pomocné třídy. Jedná se o třídy `Help` a `DraggableExtender`. Aby nebyla uživatelská hesla uložena v podobě čistého textu, můžeme z hesel vytvořit hash pomocí knihovny `BCrypt`. Tato knihovna má k dispozici sadu metod pro vytváření hashů ze zadaného textu. Aplikace používá metody `HashPassword` a `Verify`. Povinný parametr těchto metod je pouze samotné heslo. V případě první metody je výstupem hash, který je uložen do databáze. V druhém případě je výstupem pravdivostní hodnota, která určuje shodu zadaného hesla z uloženým hashem v databázi.

Třída `Help` je zodpovědná za nápovědu a animaci příslušných objektů hráči. V konstruktoru této třídy se přihlásíme k události `ScrollChanged` z komponenty `ClothRotator`. Díky tomu přesně víme, kdy byla nabídka s oblečením zastavena. Od této doby je měřen časový interval v události `Tick`. Po uplynutí stanoveného času je zavolána metoda `FindMatches`, která vyhledá z viditelných objektů nabídky ty, které patří na hrací plochu. V případě, že jsou zobrazeny tyto objekty v nabídce, metoda vrací list objektů typu `ClothItem`. Následně v metodě `Animate` vybereme náhodně jeden z nich a aplikujeme na něj animaci.

Třída `DraggableExtender` slouží k vytvoření takzvané Dependency Property, díky které jsme schopní nastavovat objektu určité další vlastnosti. V tomto případě se jedná o vlastnost `CanDrag`, díky které můžeme s objektem manipulovat metodou táhni a pusť. Toho je využito v případě ovládání pomocí myši u jednotlivých obrázků.

5.10.1 Rozpoznávání gest

Největší problém, s kterým jsem se potýkal během vývoje aplikace, bylo navrhnout, jakým způsobem bude aplikační logika rozeznávat sevřenou a zavřenou pěst uživatele, jelikož Microsoft Kinect SDK v době prvotního vývoje tuto funkcionalitu neposkytoval, musel jsem přijít s vlastním řešením. Pokud se v zorném poli nachází více uživatelů, je zajištěno zafixování na nejbližšího z nich pomocí metody `GetPrimarySkeleton`, která

jako vstupní parametr přebírá pole objektů typu `Skeleton`. Uvnitř této metody proběhne porovnání Z-ových souřadnic jednotlivých koster a je nám vrácen objekt `Skeleton`, který je k senzoru nejblíže. Tato metoda je volána podle snímkovací frekvence, tudíž máme vždy aktuální informaci o tom, s kterými daty máme dále pracovat. Stav otevřené a zavřené pěsti můžeme detekovat pomocí několika způsobů.

Neuronová síť

Rozpoznávání tohoto gesta může být realizováno pomocí neuronové sítě, která slouží jako klasifikátor [13]. Fungování této sítě je založeno na dvou hlavních fázích. V první fázi je neuronová síť ve stavu učení. V tomto případě by to znamenalo dodat síti dostatečné množství fotografií zavřené a otevřené ruky. Ke každému obrázku je přidružena informace o tom, zda je pěst otevřená či zavřená. Tím si síť vybuduje rozhodovací strom. Ve druhé fázi jsou síti předloženy obrázky rukou, a ta musí rozhodnout, zda je pěst zavřená či otevřená. Aby ale byla detekce spolehlivá z klasického RGB obrazu, museli bychom zajistit dobrou světelnost. Proto jsem se zaměřil na zpracování dat z hloubkové kamery. Vybudování kvalitní neuronové sítě nemusí být vždy úplně jednoduché, a proto jsem se rozhodl vyzkoušet jiná řešení.

Metoda založená na ploše ruky

Jinou podstatně jednodušší variantou může být využití obsahu ruky, jako informaci o tom, je-li pěst uživatele zavřená nebo otevřená. Nejprve musíme určit polohu ruky hráče v obraze. Tu získáme z objektu `Skeleton` pomocí vnitřní vlastnosti `Joints`. Tato vlastnost není nic jiného, než kolekce všech 20 jointů, které patří ke konkrétní kostře. Z této kolekce vybereme pomocí připraveného enumu `JointType` ruku, která nás zajímá. V metodě `MapSkeletonPointToDepth` poté převedeme souřadnice tohoto jointu ze skeleton streamu na hloubkový stream. Díky těmto informacím již víme, kde se ruka v hloubkovém streamu nachází.

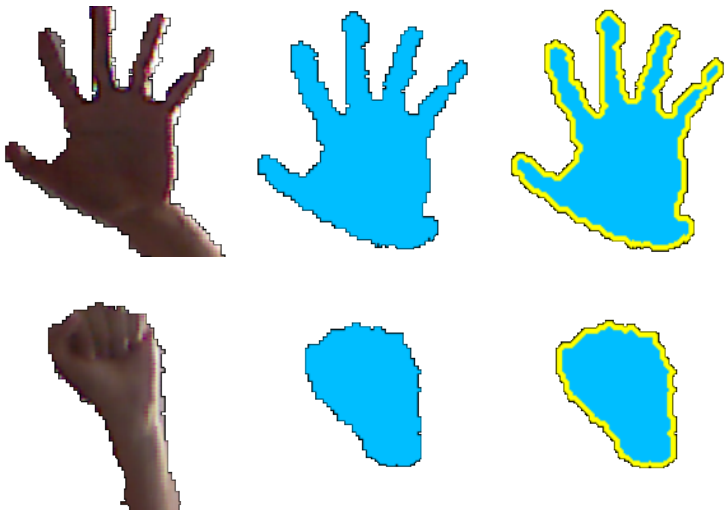
Takto získaná data využijeme jako referenční bod, který je umístěn ve středu ruky. Aby byla ruka v záběru celá, bylo nutné určit správně, v jakém okolí se pixely ruky nacházejí. Hodnota byla nastavena na 100 pixelů v obou osách s ideální vzdáleností hráče 1,2 metru od senzoru. Abychom z hloubkového snímku získali pouze data patřící k dané ruce, musíme provést výřez. Pro tento účel byla vytvořena metoda `CropDepthImage`. Té předáme samotná hloubková data, velikost ořezávané oblasti a souřadnice ruky, které jsme získali z kostry. Díky hloubkovým datům můžeme nastavit maximální odchylku v milimetrech vzhledem ke středu ruky. Jako vhodná se ukázala hodnota 10 centimetrů směrem k senzoru, jelikož potřebujeme detekovat pokrčené prsty. Ve směru od senzoru je tato hodnota nastavena na 2 centimetry. Tato hodnota je schválně nižší, aby nedocházelo k zásahu předloktí do samotného obrazu. Data, která vyhovují těmto podmínkám, jsou reprezentována barevnou složkou, nevyhovující jsou reprezentována bílou barvou.

Protože je tato metoda závislá na velikosti hráčovy ruky, musíme provést jednoduchou kalibraci, která je však běžná u většiny her používajících Kinect. Kalibrace jednoduše proběhne tak, že je uživatel nejdříve vyzván k úplnému rozevření pěsti a následně k jejímu sevření. Hodnoty naměřené při kalibraci uložíme a z nich určíme prahovou hodnotu. Tuto

metodu jsem implementoval, bohužel však nepřinášela uspokojující výsledky. Důvodů je hned několik. Mezi nimi je například malý rozdíl v ploše otevřené a sevřené pěsti, určení správné prahové hodnoty, úhel ruky a nakonec započítávání nesprávné plochy, které může být způsobeno oblečením a rukávy. Velký problém nastává, když je ruka přiložena těsně před tělo. S tímto problémem se však potýká většina aplikací, které jsou ovládány pomocí Kinectu.

Metoda založená na obvodu

Myšlenka určování stavu ruky vychází z předchozí metody. Na rozdíl od plochy je ale počítán obvod ruky. Tato veličina má daleko větší rozptyl, tudíž zde již není takový problém s nastavením prahové hodnoty. K získání hloubkových dat je využito stejné metody, která je popsána v metodě výše. Na tyto data je pak aplikován algoritmus, který vyhledává přechody mezi bílou a barevnou složkou. Pixely, které mají ve svém okolí jak barevnou tak bílou složku jsou právě ty, které určují obvod ruky hráče. Tak jako předchozí metoda, vyžaduje i tato kalibraci a konstantní vzdálenost hráče od senzoru. Ke spolehlivé detekci gesta otevřené ruky je však potřeba, aby jednotlivé prsty byly v dostatečné vzdálenosti od sebe. Díky tomu je detekován přechod mezi barvami a obvod ruky vzrůstá.



Obrázek 16: Detekce otevřené a zavřené ruky

Implementace dvou výše zmíněných metod byla provedena v samostatném projektu s názvem `OpenHandRecognizer`. Metody nebyly zakomponovány do samotné hry, jelikož nepřinášely uspokojující výsledky. Výstupy obou algoritmů můžeme vidět na obrázku 16. Vlevo jsou vidět data z barevné kamery, uprostřed je výstup metody založené obsahu, vpravo pak výstup metody založené na obvodu. Vzdálenost vůči senzoru je v tomto případě jeden metr. Při použití těchto metod se vzdálenosti dvou a více metrů rapidně klesá míra detailů, protože rozlišení hloubkové mapy je pouze 640*480 bodů. Díky tomu nejsme schopni u druhé metody detekovat jednotlivé prsty, jelikož je v obraze jednoduše

téměř není vidět. Vývoj aplikace probíhal na senzoru Kinect pro Xbox, který bohužel nepodporuje blízký mód. V případě použití zařízení Kinect pro Windows by bylo možné ruku detekovat již ze vzdálenosti 40 centimetrů od senzoru a úspěšnost této metody by byla pravděpodobně daleko větší.

Použití InteractionStreamu

Od verze Kinect SDK 1.7 přibyla možnost využití dalšího streamu popsaného v kapitole 2.6. Tento stream obsahuje celou řadu informací o ruce uživatele. Princip detekce rozevřené a zavřené ruky však není znám, jelikož tento stream funguje jako černá skříňka, které pouze poskytneme potřebná data. Vzhledem k velikosti knihovny (20 MB), kterou je potřeba referencovat, se dá očekávat, že je využito právě neuronové sítě či strojového učení.

Zpracovaná data, z kterých můžeme získat informace o hráči, jsou nám vrácena pomocí metody `CopyInteractionDataTo`, která jako parametr přebírá pole hodnot typu `UserInfo`. Každý tento objekt obsahuje vnitřní proměnnou datového typu `int` s názvem `SkeletonTrackingId`. V případě, že daný objekt obsahuje data o uživateli, je tato proměnná nastavena na nenulovou hodnotu. Skeleton stream poskytuje kostry k dvěma hráčům. Abychom byli schopni použít správnou kostru, vybereme toho uživatele, jehož `SkeletonTrackingId` je shodné s tím, které jsme získali v metodě `GetPrimarySkeleton`.

Z objektu `UserInfo` si poté vybereme požadované vlastnosti o stavu ruky. V případě detekce otevřené a zavřené ruky jsou to informace `HandEventType`. Tyto stavy jsou reprezentovány pomocí enumu `InteractionHandEventType` a to konkrétně `Grip` v případě sevřené ruky a `GripRelease` v případě otevřené ruky. Na základě těchto informací poté můžeme aplikaci oznamovat, je-li ruka hráče zavřená či otevřená. Stejně jako minulé metody, i tato je náchylná k nesprávné detekci v případě, kdy má hráč oblečení s dlouhými rukávy.

5.10.2 Mapování ruky hráče

Pro přenesení polohy ruky hráče do aplikace můžeme taktéž využít data z třídy `InteractionStream`. Po vykopírování dat z tohoto streamu máme informace o hráčích zapouzdřeny ve třídě `UserInfo`. Tato třída má vnitřní vlastnost nazvanou `HandPointer`, což představuje všechny informace o ruce hráče. Pro nás jsou obzvláště důležitá data o stavu, v jakém se daná ruka nachází a její souřadnice. Pro pohyb virtuální ruky ve 2D potřebujeme hodnoty X a Y. Ty jsou v intervalu (0,1), kde 0 na ose X znamená nejlevější část snímané scény, 1 pak nejpravější část. Analogicky je tomu v případě souřadnice Y. Jelikož v rámci běhu aplikace známe její rozměry, jednoduše tyto hodnoty vynásobíme šířkou a výškou. Výsledné hodnoty pak určují přesnou polohu virtuální ruky na plátně.

5.10.3 Kolize objektů

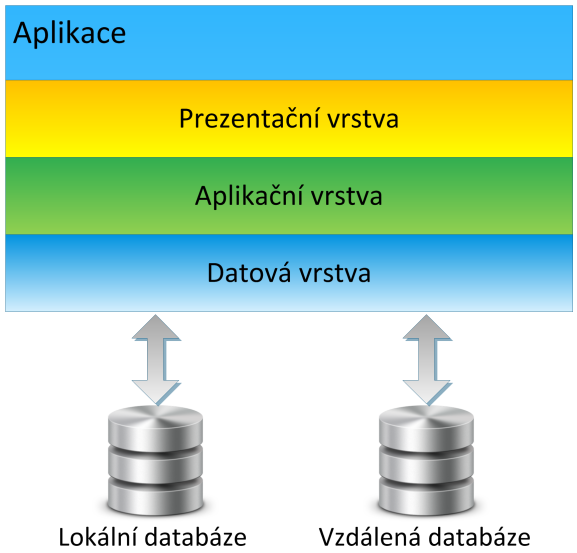
Nyní již máme všechny potřebná gesta včetně souřadnic, kde se ruka nachází. Musíme však ještě určit, zda se virtuální ruka nachází nad nějakým objektem. To je možné deteko-

vat pomocí metody `IsCollision`, která přebírá celkem tři parametry. Těmi jsou virtuální kurzor, obrázek oblečení a rodičovský kontejner společný pro oba elementy. Nejdříve si zjistíme polohu obou elementů. Dále pak z jejich velikosti spočítáme jejich šířku a výšku. Nakonec zavoláme metodu `Intersect`, která určí, zda se tyto čtverce protínají.

5.11 Datová vrstva

Všechny operace týkající se práce s databází jsou uloženy v adresáři `DressingWithKinect.Core.Network`. Ten obsahuje statickou třídu `DBConnection`, která spravuje připojovací řetězce k oběma databázím. Ty je možné nastavit v souboru App.config. Tato citlivá data jsou šifrována pomocí zabudovaných funkcí programovacího jazyka C# v metodě `CipherConnectionString`. Abychom mohli zjišťovat dostupnost vzdálené databáze, poskytuje nám třída `DBConnection` statickou metodu `IsRemoteConnection`. Po zavolání nám metoda vrací informaci o tom, zda se podařilo připojit ke vzdálené databázi. V závislosti na tom poté můžeme vybírat, do které databáze budeme přistupovat. Dále je možné zaregistrovat událost `OnInternetStatusChanged`, která nám říká, změní-li se stav připojení z online na offline a naopak.

Jak bývá dobrým zvykem, můžeme zde najít několik tříd, které reprezentují jednotlivé entity. Těmi jsou `User`, `Teacher`, `Player`, `Score` a `PlayerSettings`. Každá tato entita má vlastní DAO třídu. Tyto třídy obsahují práci s jednotlivými etitami jako je, které jsou běžně označovány CRUD.



Obrázek 17: Propojení databází s aplikací

Kritické sekce jsou opatřeny transakcemi. Mezi tyto sekce patří například načtení všech výsledků hráče, jejich vložení do vzdálené databáze a následné smazání z databáze lokální. Transakce jsou použity, aby byla zaručena konzistence dat obou databází. V případě, kdy by se podařilo načíst výsledky uživatele z lokální databáze, ale vložení by selhalo, data by byla nenávratně smazána. Data tedy můžeme bezpečně vymazat z lokální databáze až ve chvíli, kdy jsou skutečně vložena do vzdálené databáze. V případě, kdy kterýkoliv z těchto tří dílčích úkolů selže, je vyvolána aplikační výjimka. Ta je odchycena a na transakci je volán rollback, který vrátí obě databáze do původního stavu. Pokud všechny operace proběhnou bez problémů, je zavolán commit, který provedené operace potvrdí. Mezi další kritické sekce patří například vkládání uživatelského nastavení při registraci.

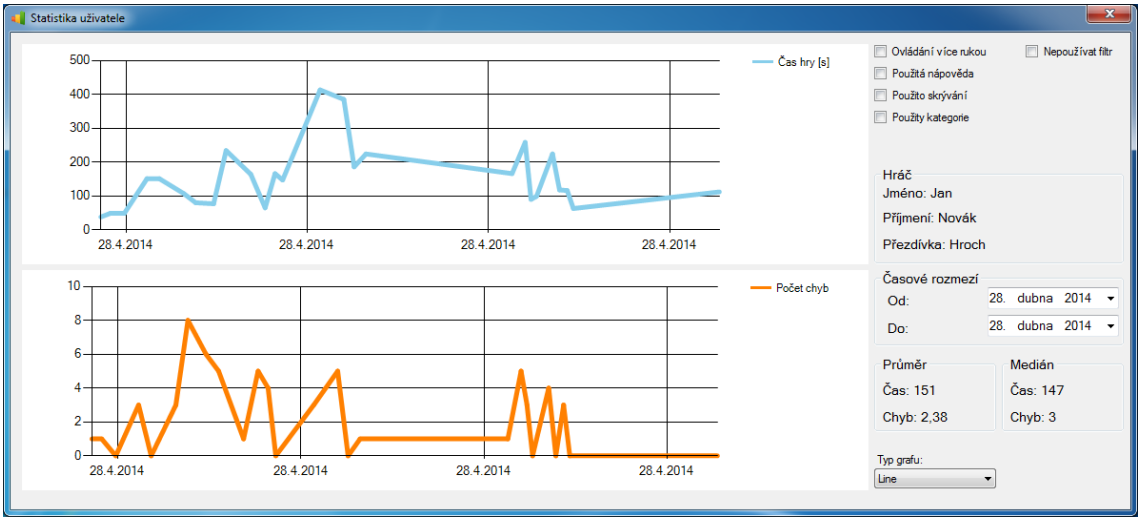
```
public static void Insert(this Player player)
{
    using (SqlConnection conn = new SqlConnection(DBConnection.Remote))
    {
        using (SqlCeConnection connCe = new SqlCeConnection(DBConnection.Local))
        {
            SqlTransaction transaction = null;
            SqlCeTransaction transactionCe = null;
            try
            {
                conn.Open();
                connCe.Open();
                transaction = conn.BeginTransaction(System.Data.IsolationLevel.Serializable);
                transactionCe = connCe.BeginTransaction(System.Data.IsolationLevel.Serializable);
                player.InsertPlayer(conn, connCe, transaction, transactionCe);
                player.InsertDefaultSettings(conn, connCe, transaction, transactionCe);
                transaction.Commit();
                transactionCe.Commit();
            }
            catch (Exception exc)
            {
                transaction.Rollback();
                transactionCe.Rollback();
                throw new Exception(exc.Message);
            }
        }
    }
}
```

Výpis 3: Vložení nového uživatele a jeho nastavení

5.12 Aplikace pro zobrazení výsledků

Pro učitele byla vytvořena aplikace, která je schopna zobrazit výsledky jednotlivých hráčů. V aplikaci je zobrazen seznam hráčů učitele, včetně podrobné statistiky jednotlivých her. Dále je možno filtrovat jednotlivé záznamy dle zvolených kritérií. Aplikace také zobrazuje průměrné dosažené hodnoty jednotlivých klientů. Díky zobrazení výsledků může učitel posoudit, jestli je hráč s postupem času schopen rozpoznávat objekty rychleji, zda klesá chybovost, či počet zapnutých asistentů. Aplikaci je možno jednoduše modifikovat a dále do ní přidávat další funkcionalitu dle potřeb.

Všechny aplikace v rámci projektu Edukin obsahovaly nástroje, které ukládaly data pro následný data mining. Takto nasbíraná data sloužila k získávání informací o pokroku klientů, vyhodnocování jejich úspěšnosti a také porovnání výsledků hráče během doby, kdy byla aplikace používána.



Obrázek 18: Aplikace pro zobrazování výsledků

6 Otestování aplikace

Abychom mohli posoudit vhodnost a stabilitu aplikace bylo nutné ji otestovat. Hra byla předvedena na dnu otevřených dveří Vysoké školy báňské v lednu 2014. Během této doby si mohli návštěvníci hru sami vyzkoušet. Díky uživatelské zpětné vazbě bylo možné opravit nedostatky a aplikaci udělat přívětivější.

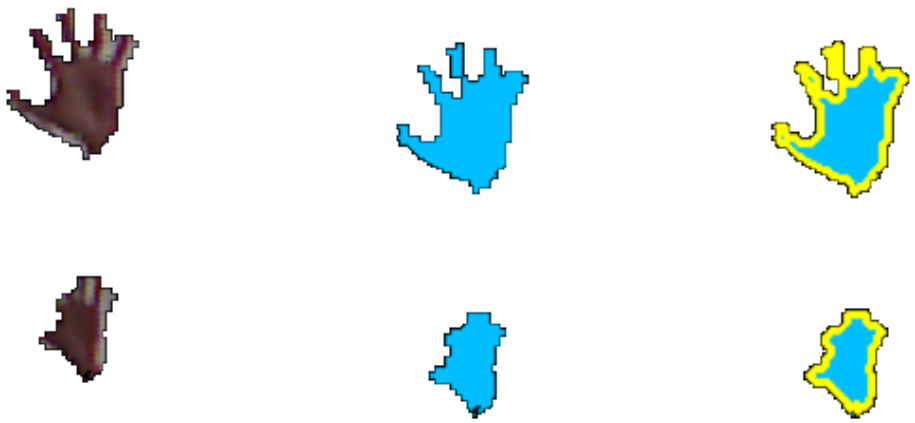
Další testování bylo provedeno na konci dubna 2014, kdy si ji mohlo zahrát přibližně 20 klientů ve stacionáři Lipová v Havířově. Pacienti aplikaci přijali velmi dobře jak po stránce individuální, tak kolektivní zábavy. Po konzultaci s vedoucím oddělení panem Mgr. Markem Zárubou byla hra posouzena jako vhodná pro rozvoj myšlení, rozpoznávacích schopností a také pro rozvoj jemné a hrubé motoriky.

6.1 Spolehlivost rozpoznávání gesta

Aby bylo samotné hraní aplikace co nejintuitivnější, bylo klíčové zvolit správnou metodu pro detekci gesta sevřené ruky. V grafech níže můžeme vidět spolehlivost jednotlivých metod pro rozpoznání gesta sevřené a otevřené ruky. Naměřené hodnoty jsou získány z 30 měření z různých vzdáleností vůči senzoru. Otestování se zúčastnily i děti, aby bylo zaručeno fungování bez problémů i při ovládání menší rukou.

Otevřenost a sevřenost ruky je však relativně dost subjektivní pojem, což může být problém. Aby byla zaručena spolehlivost, testování bylo prováděno při snímání ruky, která byla natočena v kolmém směru vůči senzoru. Další měření poté probíhalo na ruce otočená po ose Y o 90°. Prahová hodnota $(Treshold)$ pro metody pracující s obvodem a obsahem byla stanovena jako průměr počtu pixelů při zcela otevřené (O) a zavřené (C) pěsti.

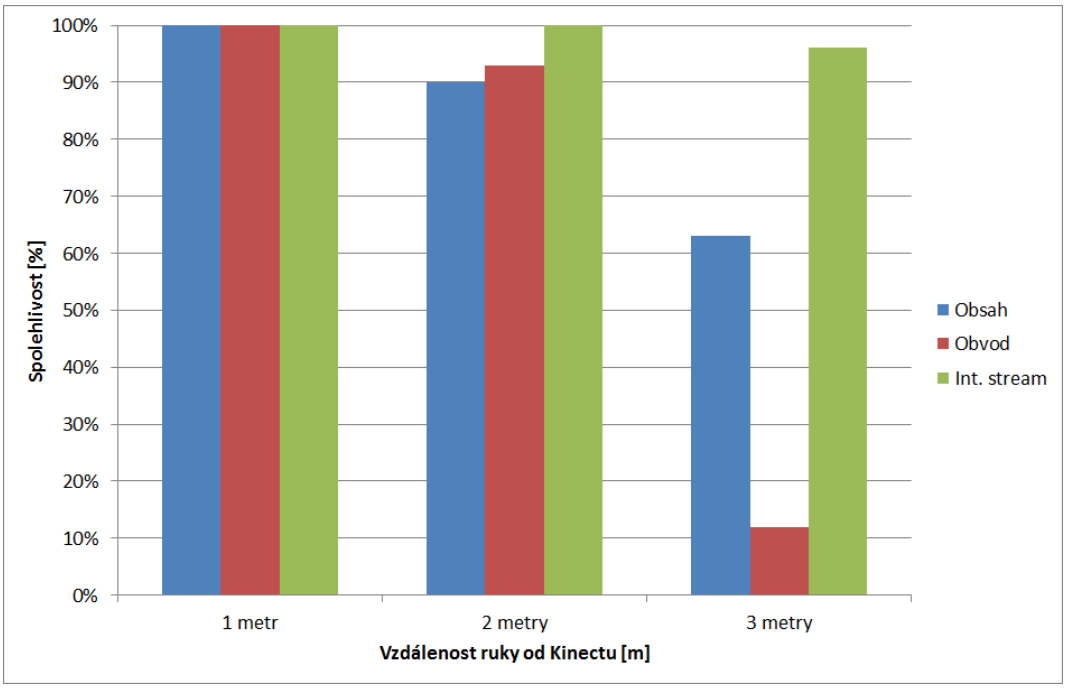
$$Treshold = (O + C)/2$$



Obrázek 19: Otevřená ruka snímaná ze vzdálenosti 2 a 3 metrů

Kolmý směr

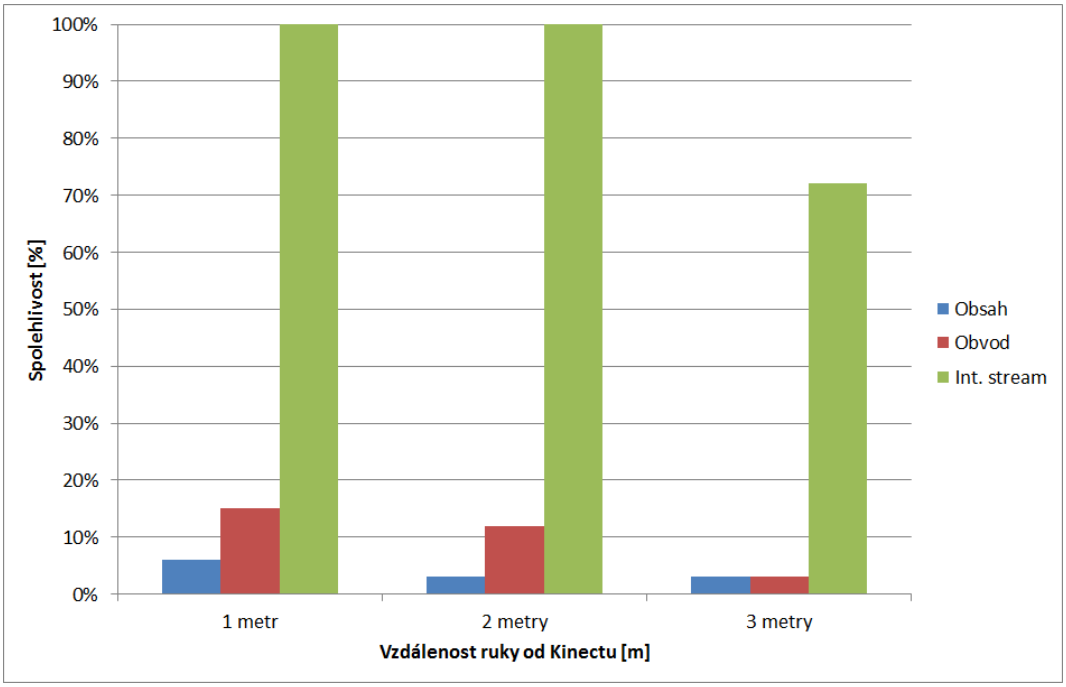
Jak můžeme vidět z grafu 20, spolehlivost metod obvodu a obsahu je dobrá pro vzdálenost do dvou metrů. Pro větší vzdálenost již zanikají jednotlivé detaily a získaná data nemají dostatečný rozptyl pro správné stanovení hraniční hodnoty. Tento problém se nejvíce týká metody obvodu, protože plocha jednotlivých prstů je velmi malá a v tak velké vzdálenosti je na hranici rozlišení snímače. Ze vzdálenosti tří metrů je již počet pixelů mapovaných na ruku okolo několika desítek, což je skutečně velmi málo. Jelikož nejsou prsty detekovány, odchylka je velmi malá a metoda selhává. Metody obvodu a obsahu dosahují největší spolehlivosti ve vzdálenosti do jeden a půl metru. V případě, kdy jsou prsty odděleny od sebe, je spolehlivost metody obvodu ještě větší.



Obrázek 20: Spolehlivost detekce v kolmém směru

Šikmý směr

Spolehlivost metod obvodu a obsahu pro jiný, než kolmý směr je špatná. Už při malých vzdálenostech tyto metody selhávají, což je jasné, jelikož pracují prahovými hodnotami vytvořenými při kalibraci v kolmém směru. I zde platí pravidlo, že pro úspěšnou detekci gesta ve všech směrech je doporučená vzdálenost maximálně dvou metrů od senzoru. Nad tyto vzdálenosti již spolehlivost klesá pod 70% i u metody, která používá `InteractionStream`.



Obrázek 21: Spolehlivost detekce v šikmém směru

U metody obvodu a obsahu nastává hned několik problémů. Jedním z nich je malé rozlišení při velké vzdálenosti uživatele od senzoru. Natočení ruky komplikuje situaci ještě více. Kvůli malému rozlišení a otáčení ruky uživatele by bylo nutné vymyslet sofistikovanější metodu pro detekci gesta toho typu. Díky jednoznačným výsledkům testů byla pro ovládání v režimu jedné ruky vybrána nejspolehlivější metoda používající `InteractionStream`.

S rozpoznáním gesta při použití obou rukou nebyl žádný problém. Spolehlivost dat z jednotlivých bodů kostry je indikována pomocí stavových informací. Bod může být v nezaměřeném, rušeném nebo zaměřeném stavu. Je-li bod v rušeném stavu, jsou jeho souřadnice dopočítány z okolních bodů a spolehlivost je relativně malá. Při zaměřeném stavu jsou hodnoty poskytované Kinectem přesné a určení polohy je velmi spolehlivé. Rozpoznání tohoto gesta bylo bezproblémové až do maximální vzdálenosti, ve které je Kinect schopen pracovat.

7 Závěr

Aplikace je dokončena dle požadovaných specifikací a je součástí projektu Edukin. Z hotových frameworků nebyly použity žádné knihovny, jelikož pro fungování aplikace nejsou zapotřebí. Dalším úkolem bylo vybudovat vlastní zjednodušenou infrastrukturu pro tuto aplikaci, která slouží pro ukládání výsledků a správu uživatelských účtů, protože stávající řešení je zbytečně komplikované a prakticky se nepoužívá. Pro účely aplikace bylo nutné navrhnout i několik jednoduchých rolí, které jsou v aplikaci potřebné pro použití v prostředí s hendikepovanými osobami. Do budoucna se počítá s vytvořením aplikace pro jednoduchou správu účtů, stejně jako s rozšířením o další potřebné scénáře a role.

Ve hře je nyní dostupných celkem 20 různých kusů oblečení. Dále se plánuje možnost přidávání dalších kusů oblečení za pomocí jednoduchých šablon. Tyto šablony by mohly být součástí jiné aplikace (například tabletové), ve které by bylo možné oblečení vytvářet a následně importovat do této hry. V případě potřeby je možné hru rozšířit o další herní módy na základě požadavků jednotlivých zařízení.

Aplikace byla otestována na dni otevřených dveří Vysoké školy báňské a poté v denním sanitáři Lipová v Havířově, kde byla pacienty velmi dobře přijata. Aby bylo možné sledovat pokroky jednotlivých klientů, je pro pracovníky v centrech připravena aplikace na zobrazování dosažených výsledků. Během této práce byla vytvořena sada nových metod pro detekci gest sevřené a otevřené ruky. Nejspolehlivější z těchto metod byla použita, aby zaručila potřebný uživatelský komfort.

Díky této diplomové práci jsem získal spoustu nových znalostí z oblasti Windows Presentation Foundation, ve kterém jsem dosud nevyvíjel žádnou aplikaci. Mnoho znalostí jsem také získal z oblasti aplikačního rozhraní samotného Kinectu a analýzy obrazu. Nejtěžší na celé práci bylo vymyslet způsob detekce gest, které nejsou součástí základního SDK. Vytvořené metody rozpoznání sevřené a otevřené pěsti byly úspěšné pouze při relativně malé vzdálenosti a kolmém směru ruky vůči senzoru. S postupem času se podařilo využít rozhraní, které zprostředkovává potřebné informace přesně pro tento typ ovládání, díky kterému je hraní hry intuitivnější a zábavnější.

Bc. Petr Hlavinka

8 Reference

[1] Mouse - Doug Engelbart Institute [online]. [cit. 2014-01-05].
Dostupné z: http://dougengelbart.org/firsts/mouse.html

[2] Kinect Ads: You Are the Controller [online]. [cit. 2014-01-05].
Dostupné z: http://www.microsoft.com/en-us/news/features/2010/oct10/10-21kinectads.aspx

[3] Wii sales - Wikipedia [online]. [cit. 2014-01-08].
Dostupné z: http://en.wikipedia.org/wiki/Wii_sales

[4] SHOTTON, Jamie, Andrew FITZGIBBON, Mat COOK, Toby SHARP, Mark FINOCCHIO, Richard MOORE, Alex KIPMAN a Andrew BLAKE. *Real-Time Human Pose Recognition in Parts from Single Depth Images.* Microsoft Research Cambridge & Xbox Incubation, [2011]. [online]. [cit. 2013-03-10].
Dostupné z: http://research.microsoft.com/pubs/145347/BodyPartRecognition.pdf

[5] KAŠPAR, Adam, *Framework pro práci s gesty založený na technologii Kinect* Vysoká škola báňská - Technická univerzita Ostrava, 2013. Diplomová práce. Vedoucí práce Mgr. Pavla Dráždilová.

[6] BOJKO, Vojtěch. *Infrastruktura založená na Windows Azure pro projekt Edukin.* Vysoká škola báňská - Technická univerzita Ostrava, 2013. Diplomová práce. Vedoucí práce Mgr. Pavla Dráždilová.

[7] JARRETT WEBB, James Ashley. *Beginning kinect programming with the microsfot kinect SDK*. New York: Apress, 2012.

[8] SEAN KEAN, Jonathan C. *Meet the Kinect An Introduction to Programming Natural User Interfaces*. Emeryville, CA: Apress, 2011.

[9] Kinect for Windows Sensor Components and Specifications [online]. [cit. 2014-02-02].
Dostupné z: http://msdn.microsoft.com/en-us/library/jj131033.aspx

[10] Kinect System Requirements [online]. [cit. 2014-02-02].
Dostupné z: http://msdn.microsoft.com/en-us/library/hh855359.aspx

[11] Kinect for Robotics - Microsoft Robotics Blog [online]. [cit. 2014-02-16].
Dostupné z: http://blogs.msdn.com/b/msroboticsstudio/archive/2011/11/29/kinect-for-robotics.aspx

[12] How To Safely Store A Password — codahale.com [online]. [cit. 2014-02-11].
Dostupné z: http://codahale.com/how-to-safely-store-a-password/

[13] Biologické algoritmy (4) - Neuronové sítě - Root.cz [online]. [cit. 2014-03-11]. Dostupné z: http://www.root.cz/clanky/biologicke-algoritmy-4-neuronove-site

[14] Kinect Interactions with(out) WPF – Part III: Demystifying the Interaction Stream - VBandi's blog - Dotneteers.net [online]. [cit. 2014-02-10]. Dostupné z: http://dotneteers.net/blogs/vbandi/archive/2013/05/03/kinect-interactions-with-wpf-part-iii-demystifying-the-interaction-stream.aspx